Europäisches Patentamt | European Patent Office | Office européen des brevets



# Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

**Patentanmeldung Nr. | Patent application No. | Demande de brevet n°**

03014839.9



Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 03014839.9
Demande no:

Anmeldetag:
Date of filing: 30.06.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

SAP AG
Neurottstrasse 16
69190 Walldorf
ALLEMAGNE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Configurable process scheduling

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F/

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

June 30, 2003
S 50315 GS

SAP Aktiengesellschaft

**Specification**

# Configurable Process Scheduling

The present invention relates to configurable process scheduling, and in particular to a method of configuring a business process for scheduling, and a method of scheduling a business process which is configured preferably by that method.

Furthermore, the invention relates to computer systems for performing the inventive methods, and to a computer-readable storage medium comprising program code for performing the inventive methods.

## BACKGROUND OF THE INVENTION

IT solutions of today for Supply Chain Execution have to be able to cope with business processes for the logistic fulfilment of orders, like purchase orders or sales orders. The fulfilment of orders is connected with the control and monitor of dates of specific activities within a business process. Some or all of these dates can either be given manually by data input of IT users, given via electronic data exchange by other business partners or have to be automatically determined by the IT system which is responsible for the fulfilment coordination of orders. In the latter case, a scheduling of the configurable business process has to be performed by the system by means of a program.

## SUMMARY OF THE INVENTION

This scheduling has to cope with free configuration of business processes, the configuration of dates that are connected with a business process and the configuration on how the dates can be scheduled. These scheduling procedures have to be configurable in order to be able to schedule business processes that integrate several business applications, which take part in the business process, with one standard tool. Due to the fact that the date scheduling may not depend exclusively on pure time parameters like time zone, working hours and net lead times of activities, an open framework is needed for the determination of dates which allows for integration of complex date determination algorithms that may incorporate in-depth logic and master data of specific applications that are responsible for the business process part to which the date belongs to.

The scheduling has to comprise the ability of scheduling at different precision levels, for example a second, a minute or an hour. This scheduling precision has to be configurable per individual date of a business process in order to allow for a scheduling which can be adapted to the specific needs of an individual business process.

The scheduling has to handle time zones property, especially in case of rounding dates to certain time units like a day, in order to allow for scheduling business processes which incorporate business activities that are spread all across the globe.

These and other objects are achieved by methods and systems according to the independent claims. Further embodiments are defined in the dependant claims.

In general, in one aspect, the invention provides a method of configuring a business process for scheduling,

the business process comprising a plurality of activities, each activity comprising at least one of a start date type and a stop date type; the activities being in a time relationship to each other; wherein

the business process is freely configurable with respect to the plurality of activities and with respect to the time relationships of the activities to each other.

Advantageous implementations of the invention can include one or more of the following features.

A technical ID may be associated with an activity or with a date type.

A text may be associated with an activity or with a date type, the text being descriptive for the activity or for the date type.

Time units may be assigned to specific date types, the time units being freely configurable for each date type.

An activity can be modeled as a plurality of sub-processes.

A sub-process may comprise a plurality of activities.

A decision whether or not a delegation may be invoked is during run-time of the scheduling.

The service functions may be usable for determination of time zone, calendar and duration of an activity.

At least one service function may be assigned to at least one activity, the service function being usable, during scheduling, for determining start date and/ or finish date of the at least one activity.

At least one delegation scheme is assigned to at least one activity, the delegation the service function being usable for invoking, during scheduling, an external application for determining start date and/ or finish date of the at least one activity.

The activities and their time relationship may be representable as a network of nodes and edges, each node representing one of the pluralities of activities, and each edge connecting a pair of nodes and representing a predecessor-successor relationship of the activities represented by the respective pair of nodes.

A scheduling scheme may be produced based on the configured business process, whereby the scheduling scheme is a set of meta data descriptive of how the individual activities are to be processed within scheduling.

A scheduling scheme may be associated to the business process, the scheduling scheme comprising configuration data to at least one of duration, calendar, and time zone.

A scheduling scheme may be associated to the business process, the scheduling scheme comprising configuration data to at least one of service function, and delegation process model.

The invention also provides a business process configured with the method preferably according to the above method.

The invention further comprises a method of configuring a production process for simulating,

the production process comprising a plurality of steps, each step comprising at least one of a start date type and a stop date type; the steps being in a time relationship to each other; wherein

the production process is freely configurable with respect to the plurality of steps and with respect to the time relationships of the steps to each other.

Also provided by the invention is a computer system for performing the inventive methods, and furthermore, a computer-readable storage medium comprising program code for performing the inventive methods when loaded into a computer system.

A schedule tool, in the following referred to as "Configurable Process Scheduling" (CPS) is developed based on Advanced Business Application Programming Object-Oriented (ABAP-OO) technique. ABAP is provided by SAP AG. CPS comprises the ability to

- define business processes consisting of activities relevant for scheduling. The business process modelling comprises the ability of top-down modelling, that is that a single activity may contain a sub-process. These sub-processes may be expanded during runtime of the CPS based on configuration settings;
- define the technical ID and language dependant short text for the activities and the dates of the activities;
- define the time units (like minute or hour) that shall be used for individual dates. Based on the given time unit, CPS performs a rounding in the frame of the local time zone of the activity,
- setup of a scheduling scheme which is a meta-data framework that holds configuration data for several service functions performed within the CPS processing. These service functions are for example used for master data determination, like time zone, calendar and duration of an activity;

- integrate other business applications by means of a delegation principle. CPS may delegate certain tasks to ABAP-OO classes, which may belong to other business components. This delegation principle can especially be used in the scheduling calculation itself when a pure generic scheduling, based on calendar, time zone and lead times is not sufficient. For example this is the case when a scheduled date has to be confirmed against capacities of resources within a complex planning application.

CPS provides a generic framework for scheduling pre-defined business processes in an environment of several business software components. CPS makes use of a process definition created by a business process modeller (BPM) and a set of meta data, the scheduling scheme, which is used to define the determination of several procedures that are performed within the scheduling. For example, the meta data of the scheme may define an access sequence to several services which may be used to provide master data needed for scheduling (duration, calendar, time zone).

The framework of CPS technically makes use of dynamic calls to ABAP-OO classes that implement certain interfaces defined by CPS. This technique is the same as the SAP Business Add-In (BADI) technique. The core framework of CPS does not contain those ABAP-OO classes that may be used within the definition of the scheduling scheme. The standard case will be that implementation of those classes and definition/shipping of schemes which make use of them has to be done by applications that make use of CPS (such as Fulfilment Coordination). Nevertheless, extensions of the core framework of CPS can be done in order to create ABAP-OO classes that are shipped by the package of CPS and can be used as elements of the definition of application-specific schemes. This may include classes that access a master data manager (MDM) or that access master data which is defined by means of CPS itself.

Given the fact that the development presented here is a generic framework, it may be used by several other applications in addition to Fulfilment Coordination. SAP development is able to deliver content within the scheduling framework (by delivering customizing data of CPS), which allows for scheduling application scenarios or business processes that are predefined by SAP Aktiengesellschaft, Walldorf, Germany. In addition, the CPS framework allows for custom specific changes of the framework content delivered by SAP and even allows for new custom specific definitions.

Configurable Process Scheduling is able to schedule a network of activities. The definition of this network consists of a description of a single activity and a description of the network.

A single activity consists of the following:

- Activity type
- Activity type category
- Date type of the start of the activity
- Date type of the end of the activity
- Calendar and working hours of the activity
- Time zone of the activity
- Time unit of the start date, time unit of the duration and time unit of the end date

An activity network consists of:

- Nodes (equivalent to a scheduling activity)
- Lines (link predecessor and successor nodes)

Free configuration of the business processes, of the dates within a business process and of the date determination procedures are not possible with Transportation and Shipment Scheduling of SAP's Advanced Planer and Optimizer (APO). Transportation and Shipment Scheduling of SAP APO performs a scheduling based on a precision of a second only. It does not comprise the possibility to configure the usage of certain time units, like minute or hour.

The invention can be used for the scheduling of dates according to a given configuration of a business process, the dates of the business process and the scheduling procedures of these dates. The invention can be used for scheduling dates of applications that are technically installed on the same R/3 system as the scheduling configuration is situated. In addition, the invention can also be used as a remote scheduling tool by applications that run on different systems. The invention can also be used independently of any business application as a stand alone tool for scheduling dates. The latter one is made possible by a user interface (UI) of the invention which allows for scheduling dates according to data given via this UI, like the lead times of activities. This UI can be accessed by a WEB browser and thus may serve as an entry point for a WEB service that schedules dates of a business process.

## BRIEF DESCRIPTION OF DRAWINGS

| | |
|---|---|
| Figures 1A-C | show the description of a scheduling activity and an activity network; |
| Figure 2 | shows in a block diagram an overview of the data storage places of the scheduling; |
| Figure 3 | shows, in a block diagram, an overview of the active components of the scheduling; |

| | |
|---|---|
| Figure 4 | shows, in a block diagram, an overview of active components of the scheduling when the explanation service is processed; |
| Figure 5 | shows, in a block diagram, an overview of active components of the scheduling when the post service is processed; |
| Figure 6A | shows the static part of the scheduling scheme; |
| Figure 6B | shows the dynamic part; |
| Figures 7A, 7B | show flow charts of the scheduling algorithm according to the invention; |
| Figure 8A-E | give examples and screen shots of several user interfaces; |
| Figure 9 | shows the definition of the business process, which is used by ICH; |
| Figure 10 | shows the assignment of time durations to specific activities; and |
| Figure 11 | shows the overview of the complete customizing of the configurable process scheduling; |

**DETAILED DESCRIPTION**

Figures 1A, 1B show the description of a scheduling activity and an activity network, respectively. It should be noted that the activity type category is referred to as 'duration type' in Figure 1A.

CPS can describe a network of activities in time by means of two different timetable formats. The complete timetable format consists of two dates for each activity (start and end-date of the activity). The derived timetable format consists of a list of dates that can be derived from the complete timetable. Figure 1C shows an example of a simple activity graph with a complete and a derived timetable. It

should be noted that in the Figures the complete and derived timetable are called extended and condensed timetable, respectively.

The scheduling comprises a data model to define the scheduling scheme, which is a set of meta data that describes a framework used for the technical processing of the scheduling. In addition to the scheme definition managed by CPS an activity network definition has to be maintained using SAP WEBFLOW builder as a business process modeller (BPM). Thus, the activity network definition is managed by the SAP WEBFLOW data model.

CPS comprises a data model of a Process Scheduling Calendar, which is the definition of working hours with a precision of a second. In addition, public holiday and factory calendars of SAP Basis, which describe the working periods with a precision of a day, can be used within CPS. CPS comprises also the possibility to make use of time zones and time units, like minute or hour, which can be defined and maintained by means of SAP Basis. CPS includes a data model of the so-called 'time unit assignment set', which assigns an individual time unit to each date of a list of dates. Thus the same business process may be scheduled with different precisions by means of using different time unit assignment sets.

CPS comprises a data model for assigning calendar, time duration and time zone to the activities of a scheduling scheme. This enables for usage of the CPS without any other source of scheduling master data. Thus, CPS may be used as a self-contained scheduling tool. In addition to this CPS master data assignment, classes that are set up in the dynamic part of the scheme (and that implement the interfaces specified by CPS) may use any additional master data available to the business software components to which these implementation classes belong.

In addition to the customizing data of the scheduling framework and the scheduling master data, CPS has to manage transactional data. This includes the data that is needed for the explanation service of CPS. This explanation data consists of a data buffer (an object in the memory) and persistent data (by means of database tables). An explanation of previously processed scheduling calls is thus made possible within a transactional context and also offline (without the transactional context in which the scheduling call was processed).

Figure 2 shows in a block diagram an overview of the data storage places of the scheduling.

CPS comprises services for the following:

- Scheduling of a business process. The scheduling service has the ability to schedule multiple independent scheduling requests in a single service call.
- Presenting and explaining the scheduling result.
- Posting scheduling transaction data and data of delegation processing.

These services are implemented as methods of a global ABAP-OO class, which may be used by any application that is situated on the same system as the scheduling framework. In addition, the services are implemented as remote enabled function modules (in accordance to the SAP BAPI standards), which might be used by any remote SAP application.

Note that the block diagrams in the following sections are in accordance with the graphic and syntax standards as described in SAPNet. The wording used in the text is also agrees with this syntax. Do not misunderstand the word 'agent'. An

'agent' in this context is a piece of a functional program, such as an ABAP-OO class method or a function module, in contrast to other types of building blocks such as data storage places.

Figure 3 shows, in a block diagram, an overview of the active components of the scheduling when the scheduling service is processed. Please note, that in the Figure, CPS is called 'SCM scheduling'.

An external application calls the scheduling service via its application interface (API). The CPS ('SCM scheduling') first routes all methods of the API through an entry agent which then carries out a functional dispatch. This entry agent provides a single point of entry to the CPS framework internally, although the framework provides several API methods.

If scheduling is requested, the entry agent will dispatch to the scheduling controller. The controller first performs a pre-step that makes use of the scheduling toolkit in order to derive the static scheme and the dynamic scheme. This results in a description of the complete graph of activities that includes information from the scheduling scheme, the activity network and the master data. This graph of activities description holds all information necessary for a subsequent scheduling engine. The controller then invokes the scheduling engine.

The scheduling engine performs the pure scheduling algorithm for a network of activities.

Scheduling of individual activities is done by the activity objects that are instances of an ABAP-OO class. Activity objects are contained in the description of the graph of activities built by the controller. They make use of an interface of a time stream agent that is able to carry out basic scheduling functions by means of SAP

Basis package SZTI. Package SZTI provides the functions needed to calculate with different time units taking time zones, factory calendars and time streams into consideration. Activity objects also make use of an interface for external processing, the so-called delegation, which might have to be called for certain activities in order to set start- or end-date of the activity, or in order to schedule the complete activity using an external method.

After the scheduling engine has completed its work, the controller stores the result and additional explanation data in the explanation data buffer.

Figure 4 shows, in a block diagram, an overview of active components of the scheduling when the explanation service is processed: Please note, that in the again, CPS is called 'SCM scheduling'.

An external application calls the scheduling service via its API. CPS ('SCM scheduling') first routes all methods of the API through an entry agent which then carries out a functional dispatch. If an explanation is requested, the entry agent dispatcher will call the explanation agent.

The explanation agent first retrieves data of a preceding scheduling service call (which may still be stored in the transient explanation data buffer or which is available via persistent database storage) and then provides several views. The explanation agent contains a view controller in order to manage the different views. Views to explain the results of delegation ('external processing') can also be provided.

Figure 5 shows, in a block diagram, an overview of active components of the scheduling when the post service is processed. Please note, that in the figure CPS

is called 'SCM scheduling'. An external application calls the scheduling service via its API. CPS ('SCM scheduling') first routes all methods of the API through an entry agent which then carries out a functional dispatch.

If a post is requested, the entry agent dispatcher will call the post agent. The post agent may retrieve the transient explanation data buffer and post this data to the database.

The post agent will raise a public event 'POST' of ABAP-OO class /SCMB/CL_SC_CONT. This enables all instances of classes that were previously used for delegation ('external processing') to execute their specific postings (data saving). These classes will not be destroyed when the scheduling method is carried out if they subscribe to this event. Thus CPS comprises a transactional concept, which ensures consistent data postings across several business applications which might take part in the scheduling (via the delegation principle).

A scheduling scheme is a set of meta-data that describes a framework that is used for the technical processing of the scheduling. It is a definition on how to handle individual activities of business processes within the scheduling and it is a definition on how to determine the master data needed for scheduling, such as duration and calendar of an activity.

The data model of the scheduling scheme consists of the definition of 'atomic' objects (which are the components of a scheme) and the definition of the scheme itself. 'Atomic objects' may be reused in several different schemes. 'Atomic objects' are:

- Activity type (like 'Load at issuing plant' or 'Load ship at harbor', for example): Defines the activities that can be used to define scheduling schemes. For

each activity an ABAP-OO class can be maintained as default at the activity level for external processing. Activity types are defined by tables /SCMB/TSCACTI and /SCMB/TSCACTIT (see Figures 6A, B).

- Activity type category (like 'loading', 'packing', 'picking'): Several activities of a scheduling scheme may use the same duration type. For example 'loading' might be used for an activity at the issuing plant but may be used in addition for an activity at a load transfer point (at a harbor ...). Based on the activity type category conditions/rules for determination of the duration of an activity may be setup. Activity type categories are defined by tables /SCMB/TSCDURA and /SCMB/TSCDURAT (see Figures 6A, B).

- Date type (like 'delivery date', 'material availability date'): Date types are used to describe the start and end dates of an activity. This is the so-called complete timetable format. Date types may also be used to define the derived timetable format. The derived format contains a subset of all start/end dates of activities of the process. A date type may be used for the complete and for the derived timetable format at the same time. The date type does not carry a category that differentiates between the complete/derived time table formats. Date types are defined by tables /SCMB/TSCDATE and /SCMB/TSCDATET (see Figures 6A, B).

Each of the 'atomic objects' has a technical name and a language-dependent text that can be used for a UI presentation.

A scheduling scheme comprises a list of activity types for which the framework definition has been set up. Only activity types that belong to the same scheme can be used in the definition of a business process (one process definition of the BPM can refer to a single scheme only). A scheme definition can be reused in several different business process definitions, with the restriction, that the business processes use only those activity types that belong to the scheme.

Scheduling schemes have to be pre-defined during a setup phase when the business processes, that is, activity network, are defined. Applications that make use of the scheduling service in order to schedule individual business processes pass the key of the activity network definition (that is, workflow-ID or a process alias name that is connected to the workflow-ID via a mapping available in the CPS customizing) to the scheduling. This activity network defines which activities are present in the business process and how they are related in time. In addition, the activity network definition contains a reference to the scheme that will be used in the scheduling.

The 'atomic objects' and the scheme will have a protected SAP namespace by using table TRESC. This allows the delivery of standard business scenarios by delivering table datasets that describe standard atomic objects and standard schemes. Note that the namespace definition of table TRESC is the major reason for the existence of the symbolic names for ABAP-OO classes that will be used in the CPS framework.

A scheduling scheme consists of a static part of the scheme and a dynamic part of a scheme. Figure 6A shows the static part of the scheduling scheme, while Figure 6B shows the dynamic part.

Definition of the static part of scheme (see Fig. 6A):

- Table /SCMB/TSCSCHM defines the scheme identifiers.
- Table /SCMB/TSCSCHMT provides a language-dependent description text for each scheme identifier.
- A scheme consists of a list of activity types (see table /SCMB/TSCSCHE)

- For each activity type of a scheme the activity type category, date type of start and end of the activity has to be defined. This is done by the fields DURATYPE, SDATETYPE and EDATETYPE of table /SCMB/TSCSCHE. Note the restriction that the date types used for start/end of the activities have to be unique within a scheme. This allows for unambiguous transformation between date types and start/end-points of activities. The uniqueness of the date types will be guaranteed by the secondary indexes of table /SCMB/TSCSCHE and by checking the input data in the view maintenance.

- For each activity of a scheme an ABAP-OO class for external processing can be set as default at the level scheme/activity (field CLASS_NAME in table /SCMB/TSCSCHE). The ABAP-OO class is set by a symbolic name CLASS_NAME. The symbolic names CLASS_NAME are defined in the dynamic part of the scheme using table /SCMB/TSCCLASS.

- For each activity of a scheme a control parameter that enables for delegation ('external processing') can be maintained (field CLASS_USAGE of table /SCMB/TSCSCHE). This parameter can have the values: No external method class use, always take default from activity definition, always take default from definition of static scheme, always take dynamic determination depending on application data, evaluation by priority: Dynamic determination, static scheme definition, activity definition

- For each activity of a scheme the start and end can be mapped to a date type that is used for the derived timetable format. This mapping is done using table /SCMB/TSCSCHD. Field DATETYPE denotes a date type of the derived timetable format. ACTITYPE denotes an activity type of the scheme and ACTIDCAT denotes whether the start or end of the activity should be mapped (ACTIDCAT may have the values A = 'start of activities', B = 'end of activity').

The dynamic part of a scheme (see Fig. 6B) defines the behaviour of the scheduling for a certain scheme depending on application data that is passed to the scheduling service call. The definition of the dynamic part comprises:

- A definition for ABAP-OO classes (via table /SCMB/TSCCLASS) that can be used in the scheme definition. This definition indicates for which purpose the class may be used. This is indicated by the fields USE_* which are simple flags. The definition is done by first defining a symbolic name for an ABAP-OO class (field CLASS_NAME) and by linking an ABAP-OO class to this symbolic name (field SEOCLSNAME). It is thus possible to ship the datasets of table /SCMB/TSCCLASS as predefined SAP customizing content (like for all other CPS customizing tables). In addition, using table /SCMB/TSCCLASST, a text describing the functional role of the class can be defined for each symbolic class name.

- A definition of the transformation of externally provided properties (passed to the scheduling via an interface) into the internally used table of properties. This is done using table /SCMB/TSCIMAP that might have several datasets for an internal property DATA_NAME. Mapping might be carried out directly by passing the value of a property PROP_NAME (provided by the calling application via the interface of the scheduling) or it might be performed in a more complex way using an ABAP-OO class that implements the mapping interface /SCMB/IF_SC_IMAP. In addition to the mapping, all properties that are passed via CPS interface are transferred with a name identical to an internally used property. This is only true for those externally provided property names that are not used as DATA_NAME in table /SCMB/TSCIMAP. The internally used table of properties is the starting point for the determination procedures described below.

- A definition of location/business partner determination. This is done using table /SCMB/TSCLOCA. For each start/end date of an activity of a scheme a location/partner may be determined directly by value mapping from an internal property DATA_NAME or by using an ABAP-OO class that implements

interface /SCMB/IF_SC_LOCA. These locations/partner IDs may subsequently be used to determine the duration, working hours and time zone of the activity. Within the scheduling framework several different types of business partners/locations may be used. For example one may make use of the ABA business partner with ABAP data element BU_PARTNER (CHAR10) or the APO location with ABAP data element /SAPAPO/LOCID (CHAR22). Inside the scheduling framework the partner will be treated in a generic manner as a property of the activity. Applications that make use of CPS are responsible for the consistency of the partner determination setup and the procedures setup (see below) that may make use of these partner IDs. The procedures that use the partner IDs have to be adapted to the type of partner.

- A definition of enrichment of internally used property data. This is done using table /SCMB/TSCMAST. For each scheme a set of internal property names (field DATA_NAME) can be defined. For each property name several ABAP-OO classes, which implement interface /SCMB/IF_SC_MAST, can be specified with different priorities. In case the property is not yet known in the scheduling (because it could not be determined by transformation from the properties provided via the scheduling interface), CPS will use the ABAP-OO classes in order to determine the property. This is the how master data or application- specific data might be read during Configurable Process Scheduling. Access to the data is not performed in the core coding of the CPS framework but is done within the dynamically chosen ABAP-OO classes. These implementation classes may belong to a package/software component different to CPS.

- A definition of the working hour (time stream) determination. This is done using table /SCMB/TSCTSTR. For each activity of a scheme the time stream id might be determined by directly passing the value of an internal property or it might be determined by means of an ABAP-OO class that implements interface /SCMB/IF_SC_TSTR. In order to schedule with the time unit 'day' in addition to a time stream, a factory calendar may be determined by means of

the ABAP-OO class. It is no be noted that the basic scheduling algorithm decides on the basis of the given time unit of the activity duration whether a SAP time stream, a SAP factory calendar or no work time description is used. For time units below a day, always the time stream is used which may describe working hours with a precision of a second. In case of an activity duration time unit of a day, a SAP factory calendar is used. In case of duration time units greater than a day, like a week or a month, no work time description is used. The value of a property may consist of a time stream ID concatenated by a factory calendar ID. A factory calendar can also be determined by means of a property. For an activity of a scheme several datasets of table /SCMB/TSCTSTR might exist with different priorities.

- A definition of duration determination. This is done using table /SCMB/TSCDURG and is analogous to time stream determination. Note that duration consists of a value and a unit. The value of the internal property that is used for duration determination has to consist of a value concatenated with a unit.

- A definition of time zone determination of activities. This is done using table /SCMB/TSCTZON and analogous to time stream determination. The time zone of activities is needed to round dates and to schedule in case a factory calendar is used. Note that each activity has a single time zone. The time zones used in the scheduling algorithm to round start and end dates are identical to the time zone used to schedule the duration with regards to a calendar. If an application wants to make use of different time zones, it has to model an activity network consisting of several activities. For example a transport from Germany to US-East that will include the two different time zones has to be modelled by three activities: Goods issue, transport and goods receipt. Goods issue and goods receipt will have the time zones CET and EST, whereas the transport activity will have the time zone that corresponds to the working calendar of the ship.

- A definition of the determination of ABAP-OO classes for delegation (also called 'external processing'). This is done by table /SCMB/TSCMETG and is analogous to time stream determination. This determination may take place depending on the CLASS_USAGE field of table /SCMB/TSCSCHE.
- A definition on how to evaluate if sub-networks, which are attached to an activity, will be expanded in the scheduling. This is done by table /SCMB/TSCEXPN.

Note that the four identical-looking tables /SCMB/TSCTSTR, /SCMB/TSCTZON, /SCMB/TSCDURG, /SCMB/TSCEXPN will differ by the ABAP data elements used for the field CLASS_NAME. Thus, dedicated search help and documentation can be assigned to each use case of the classes.

With the definition of the activity network a business process is defined from point of view of the scheduling. The activity network defines the set of activities present in a business process as well as their relation in time. The network definition has to include a cross-reference to a scheduling scheme that will be used during the scheduling of the activity net.

The network definition is carried out by means of the SAP WEBFLOW (SAP workflow). Configurable Process Scheduling reads the data stored within SAP WEBFLOW. In the SAP WEBFLOW business process modeller (BPM) in the basic data section, which is the data at header level of a workflow, a property that denotes the key of a scheme has to be defined. The name of the property will be a fixed name 'SAP.SCM.BAS.SCH.SCHEME'. With this property the scheme that will be used in the scheduling of the process is denoted.

In the step maintenance of the BPM, you can define that a property that denotes the technical name of the business process step is a scheduling activity. The name of the property will be a fixed name 'SAP.SCM.BAS.SCH.ACTITYPE'. If this property is present for a workflow step, this step is relevant for scheduling. If this property is not present for a workflow step the step is not relevant for scheduling. The value of the property denotes the activity type of the workflow step.

A network of scheduling activities can be derived from the workflow network of steps. This network of scheduling activities may not contain all workflow steps due to neglecting steps that are irrelevant to scheduling. The set of links between scheduling activities of the network of scheduling activities is derived from the links of the workflow network of steps. Workflow lines that end at a step irrelevant to scheduling are concatenated. These concatenated lines are the lines of the activity network that have a step relevant to scheduling at both ends of the line.

The BPM of SAP workflow allows for top-down modelling of a business process that results in steps that contain a sub-network (sub flow). The header property '...SCHEME' of sub-networks has to be identical to the scheme used for the overall network definition (parent workflow). Configurable Process Scheduling will be able to expand these sub-networks in order to obtain the complete activity network. Steps that contain sub-networks do not need to have the property '...ACTITYPE'. In the latter case, these steps themselves will never be part of the scheduling network (but their sub-network will be part of the scheduling network).

Note that the BPM process definitions can be shipped by SAP development as business content. This allows the definition of standard processes by SAP development.

If the activity network definition contains process steps that are sub-networks, two different types of process steps have to be handled differently in the scheduling:

1) A process step contains a sub-network and has the property '...ACTITYPE'. In this case two possibilities exist:
   a) Expanding the sub-network during scheduling
   b) Do not expand the sub-network but take the parent step as a single scheduling activity as an estimate of the sub-network instead
2) A step that contains a sub-network does not have the property '...ACTITYPE'. In this case the sub-network always has to be expanded in the scheduling since no representation of this part of the process would be there in the scheduling otherwise.

During the scheduling of a network, a decision has to be made for steps of type 1) whether the sub-network should be expanded or not. This will be done according to the settings of table /SCMB/TSCEXPN in the same way as the other tables of the dynamic part of the scheme (like table /SCMB/TSCMETG or /SCMB/TSCTSTR). Thus the expansion of individual activities can be controlled by several different procedures (via interface data and via different ABAP-OO class implementations of an interface) with different priorities. For example, you may set up a dataset in table /SCMB/TSCEXPN for each activity with DATA_NAME ='ACTITYPE_EXPAND' where ACTITYPE is the name of the activity as given in the BPM step which holds the sub-network. Using interface data (table of properties) of the scheduling, the calling application may then control the expansion of certain sub-networks by submitting the relevant property 'ACTITYPE_EXPAND' with value 'TRUE'.

Zooming into the details of a business process and thus scheduling it by taking into account more details may be used to have different views of a business process. This could be depending on the current process step where the business process is. You might have different levels of zoom for a preview of a business process, and during the different stages when the process is partially evolved.

Note that the use of sub-networks to zoom into details of a process is limited to those sub-networks that were defined during the design phase of a business process.

Delegation ('external processing') will be possible not only for single activities but for a sub-network that comprises several activities contained in the overall network. This will be possible for activities that are directly linked together, that perform 2nd grade external processing (that is, the complete activity is determined by delegation) and all activities make use of the same ABAP-OO implementation class for delegation.

Because these ABAP-OO classes may be determined dynamically during runtime, a static definition of the sub-networks that can be processed coherently (together) is not possible. The determination of sub-networks for coherent external processing is done during runtime according to the criteria given above.

Note that these sub-networks may have several inbound links and several outbound links. They are not limited to a trivial sub-network consisting of single in- and outbound line.

Note also that the use of sub-networks for coherent external processing is not driven by the design phase of the process (it is not due to network master data). It

is due to the settings in the scheme master data that are evaluated during runtime and which describe the technical settings for processing. From a business point of view, the packaging for coherent external processing is just a technical detail; the business process itself remains unchanged.

The determination of sub-networks for coherent external processing is done in the pre-step of the controller. Thus the schedule algorithm makes use of the network definition and does not perform any network determinations itself. The overall architecture thus remains open for other input sources of the network definition.

Figures 7A, 7B show flow charts of the scheduling algorithm according to the invention. Figure 7A is a flow chart for a sequential graph. Figure 7B is a more complex graph using a depth first search (DFS) algorithm.

The following features are accounted for in the algorithm:

1) Scheduling a network of activities including time units and delegation of single dates, activities or sub-networks. The network propagation is done by a commonly known graph algorithm, the DFS algorithm.

2) Optionally take a list of additional entry point dates into account with each having a specific priority. CPS will first perform a scheduling by using the entry point with highest priority and than checks, whether the result is consistent with the value of the nominal entry point. If this is the case, the result of the first scheduling will be taken as the overall result. If this is not the case, all other entry points are used for subsequent scheduling calculations. After each individual scheduling the check is re-evaluated. This feature of additional entry points can be used for emulating roundtrips in the scheduling requests. This is needed because in general, a forward scheduling along a sequence of activities followed by a backward scheduling along the same sequence of ac-

tivities does not result in the original starting time stamp. In general, scheduling is not 'reversible'.

3) Optionally take constraint on earliest date into account. The constraint may be specified individually for each date type or globally for the complete process.

4) Use optionally push or pull optimization. In case an optimization is used, at the very end of the scheduling the network algorithm will finally starts again at the activity date given in the optimization request and propagates once more towards the future or past time direction in case of push or pull optimization respectively. This kind of optimization is very useful for specific processes. For example, in case of forward scheduling of an outbound process (like a outbound delivery for a sales order) a pull optimization with the delivery date will ensure that there do not occur unwanted time gaps between the activities of the process. These time gaps would lead to an increased overall lead time of the process and, as a consequence, to material requirement dates earlier than needed for the fulfillment of the scheduled delivery date.

5) Take date fixing into account. This allows for date enrichment of a given process by means of CPS, where some of the process dates may already be known and fixed.

6) Define a maximum number of single activity scheduling calculations depending on the given scheduling request, i.e. depending on optimization and fixation requests. Monitor the number of single activity scheduling calculations in order to recognize un-resolvable scheduling requests.

7) The resulting dates are rounded according to their given time unit in the time zone of the activity. The resulting dates are thus time slices with an inclusive start and an exclusive end time stamp. (It is to be noted that a time stamp describes a point in time with the precision of a second.

External processing may be used to:

a) Determine both start and end dates of the activity ($2^{nd}$ grade processing),

or

b) Determine the start date of the activity (1st grade processing),
or

c) Determine the end date of activity (1st grade processing).

During the pre-step of the CPS, each ABAP-OO class to be used for external processing (according to the evaluation of the dynamic scheme) is asked to determine the details for external processing. This allows the external class to specify at most one out of the three possibilities a), b) and c). This information is stored in the graph objects of configurable process scheduling (ABAP Class /SCMB/CL_SC_GRAPH) and is thus transferred to the scheduling calculator. Therefore the calculator has the information as to whether 1st grade or 2nd grade external processing will be applied.

The total runtime of CPS per individual scheduling request was measured to be in the order of 10 Milliseconds with an SAP APO4.0 system. Naturally, the runtime depends on the methods used within the scheduling framework, namely the methods used for example for master data determination in order to determine activity durations, time zones and calendars.

In the use case example of SAP distribution replenishment planning (DRP), which will use CPS with a simple business process of three sequential activities goods issue, transport and goods receipt, a runtime of 5-8 Milliseconds was measured per scheduling request within a mass planning transaction. The DRP application submits the calendar, time zone and time duration of each activity of the process via the scheduling interface. Thus within the CPS framework no complex methods for master data determination are needed.

Figure 8A shows the user interface for the definition of the elements of schemes.

Figures 8B, 8C show the definition of the logical classes which can subsequently be used within the definition of a scheduling schema. The definition of a logical class includes the definition of the possible usages of the class.

Figures 8D, 8E show the UI for the definition of scheduling schemes.

Figure 9 shows the definition of the business process, which is used by ICH and DRP for scheduling a process consisting of goods issue, transportation and goods receipt. This workflow definition is shipped by SAP within the CPS package.

Figure 10 shows the assignment of time durations to specific activities. Calendars (working hours) and time zones can be assigned analogous. This data is used within CPS in case specific settings within the scheduling scheme are used which ensure that certain ABAP-OO implementation classes, which read these data, are used for the determination of the duration of the activity.

Figure 11 shows the overview of the complete customizing of the configurable process scheduling.

The configurable process scheduling comprises a test tool (see last point in the customizing view above), which enables to perform a stand-alone scheduling outside of other applications and which results in an explanation UI that provides several views. The views according to Figures 12A, 12B provide the scheduling result as well as information on the insights of the scheduling.

Annexes 1-3 are part of the invention.

**Annex 1**

Glossary

In the following, some definitions are given.

Step of a business process:

Part of a business process that can be considered independently in the description of a business process. A business process can be described by a set of steps and the relations of these steps.

Scheduling activity:

Step of a business process that should be taken into account by scheduling because this process step needs a period of time (duration) for processing. This processing duration can either be of technical type (for example period of time needed for certain IT actions) or of business type (for example period of time needed for picking of goods in a warehouse). Note that in this document the word 'activity' always implies relevance for scheduling, otherwise the word 'step' is used.

Scheduling activity network:

Description of the relative arrangement of business process activities with regard to time. This also defines the scheduling activities that are present in the business process.

Date type:

Technical name with a (multilingual) text that describes the business content of a date, for example technical name 'LDDAT' with description 'start of loading at shipping plant'. Each activity has two date types that describe the start date and the finish date of the activity.

Date:

Consists of a date type and a value. The value can be either a time stamp or a time slice, depending on the precision of the date. The technical name of the date type identifies the date technically.

Scheduling timetable:

Set of dates that describes the business process with regard to time. The result of a scheduling service call is a scheduling timetable. Configurable Process Scheduling (CPS) may work with two different formats of timetable, the complete and the derived timetable (see below).

Complete timetable:

Contains a start date and a finish date for each activity of a scheduling activity network. Each date type can only be used once in the complete timetable.

Derived timetable:

A list of dates that can be derived from the complete timetable. For example, the derived timetable may be a subset of the dates of the complete timetable. A date from the derived timetable can be transferred to a date of the complete timetable. Dates that can be transferred from the complete timetable to the derived timetable and vice versa do not necessarily need to have the same date type. In this way the description of the business process dates can be different in the complete and derived timetable.

Scheduling schema:

A set of meta data that describes a framework used for the technical processing of scheduling. It defines how individual activities of business processes are to be handled within scheduling. A scheduling schema contains a list of those activities for which the framework is defined. A scheduling schema can be used by several different business processes (described by a scheduling activity network). The schema has to contain all scheduling activities of the process. The schema may

contain more scheduling activities than the scheduling activity network. The scheduling schema consists of a static part and a dynamic part.

Activity type:

Technical name with a (multilingual) text that describes the business content of an activity (from the point of view of scheduling). Each activity of a process has such an activity type. An activity type has to be unique within a single process. Thus the activity type identifies one step of a business process. For example an activity type may be 'LOADATPLANT' with the description text 'load at shipping plant'.

Activity type category:

Technical name with a (multilingual) text that describes the business content of an activity. Each activity of a process has an activity type category. The difference between activity type category and activity type is that an activity type category within a business process does not need to be unique. The activity type category describes the business content of an activity less precisely than the activity type. Thus, several activities that can have a very similar content (such as 'load at shipping plant' and 'load at transshipment point') can be treated in a similar way by scheduling. For example, the time needed to process the activity can be determined solely using the activity type category. (In the above example, this would mean that the duration of loading is independent of the location shipping plant/ transshipment point).

Delegation:

Delegation denotes that an activity is (partially) scheduled by means of an ABAP-OO class that does not belong to the core framework of Configurable Process Scheduling. These classes may belong to any software package that includes any other component/application.

Partial delegation:

Only one date (start or finish date) of the activity is determined.

Complete delegation:

Start and finish date of the activity are scheduled.

Coherent complete delegation:

A set of directly linked activities (a scheduling activity network) is scheduled together by a single call to a method of an ABAP-OO class. A complete scheduling result is thus obtained for all activities. The network of activities that is coherently processed can be a sub-network of the scheduling activity network or the complete scheduling activity network.

Time unit:

is needed to define dates and durations. Each duration needs a number and a time unit. A time unit is optional to describe a date with a certain precision. For example a date with the unit 'day' will not be precise at 'seconds' level, it will be rounded to a precision of a day. A date without a time unit is a simple time stamp with a precision of a second. This is also called time unit 'exact'. A date with a time unit is a time slice with a start and end time stamp.

Time unit assignments:

Defines for a set of date types a time unit that will be used in the scheduling algorithm. Time unit assignment can be re-used in several different business processes. Date types that are used in the business process but that have no time unit assigned are treated as dates with time unit 'exact'. This means that they are simple time stamps only.

Scheduling calendar:

An SAP factory calendar or an SAP time stream.

Business process modeler (BPM):

An application of SAP Basis that will be developed within Basis release 6.20/6.30. The BPM will be an enhancement of the already-existing workflow builder. Using the BPM, a business process can be defined by setting up a network of process steps.

Entry point date:

The date that is passed to CPS as the starting point for the scheduling algorithm. The date has to exist in the timetable (complete or derived) of the activity network. CPS may change this date; it is not fixed in the scheduling algorithm.

Alias of a business process:

Refers to an existing workflow definition via a mapping table. The scheduling can thus be called using an alias of a business process instead of a workflow ID.

## Annex 2

Glossary

***Step of a business process***

*Part of a business process that can be considered independently in the description of a business process. A business process can be described by a set of steps and the relations of these steps.*

***Scheduling activity***

*Step of a business process that should be taken into account by scheduling because this process step needs a period of time (duration) for processing. This processing duration can either be of technical type (for example period of time needed for certain IT actions) or of business type (for example period of time needed for picking of goods in a warehouse). Note that in this document the word 'activity' always implies relevance for scheduling, otherwise the word 'step' is used.*

***Scheduling activity network***

*Description of the relative arrangement of business process activities with regard to time. This also defines the scheduling activities that are present in the business process.*

***Date type***

*Technical name with a (multilingual) text that describes the business content of a date, for example technical name 'LDDAT' with description 'start of loading at shipping plant'. Each activity has two date types that describe the start date and the finish date of the activity.*

***Date***

*Consists of a date type and a value. The value can be either a time stamp or a time slice, depending on the precision of the date. The technical name of the date type identifies the date technically.*

***Scheduling timetable***

*Set of dates that describes the business process with regard to time. The result of an SCM scheduling service call is a scheduling timetable. SCM scheduling may work with two different formats of timetable, the complete and the derived timetable (see below).*

***Complete timetable***

*Contains a start date and a finish date for each activity of a scheduling activity network. Each date type can only be used once in the complete timetable.*

***Derived timetable***

*A list of dates that can be derived from the complete timetable. For example, the derived timetable may be a subset of the dates of the complete timetable. A date from the derived timetable can be transferred to a date of the complete timetable. Dates that can be transferred from the complete timetable to the derived timetable and vice versa do not necessarily need to have the same date type. In this way the description of the business process dates can be different in the complete and derived timetable.*

***Scheduling schema***

*A set of meta data that describes a framework used for the technical processing of scheduling. It defines how individual activities of business processes are to be handled within scheduling. A scheduling schema contains a list of those activities for which the framework is defined. A scheduling schema can be used by several different business processes (described by a scheduling activity network). The schema has to contain all scheduling activities of the process. The schema may contain more scheduling activities than the scheduling activity network. The scheduling schema consists of a static part and a dynamic part.*

***Activity type***

*Technical name with a (multilingual) text that describes the business content of an activity (from the point of view of scheduling). Each activity of a process has such an activity type. An activity type has to be unique within a single process. Thus the activity type identifies one step of a business process. For example an activity type may be 'LOADATPLANT' with the description text 'load at shipping plant'.*

***Activity type category***

*Technical name with a (multilingual) text that describes the business content of an activity. Each activity of a process has an activity type category. The difference between activity type category and activity type is that an activity type category within a business process does not need to be unique. The activity type category describes the business content of an activity less precisely than the activity type. Thus, several activities that can have a very similar content (such as 'load at shipping plant' and 'load at transshipment point') can be treated in a similar way by scheduling. For example, the time needed to process the activity can be determined solely using the activity type category. (In the above example, this would mean that the duration of loading is independent of the location shipping plant/ transshipment point).*

***Delegation***

*Delegation denotes that an activity is (partially) scheduled by means of an ABAP-OO class that does not belong to the core framework of SCM scheduling. These classes may belong to any package that includes any other component/application.*

***Partial delegation***

*Only one date (start or finish date) of the activity is determined.*

***Complete delegation***

*Start and finish date of the activity are scheduled.*

***Coherent complete delegation***

*A set of directly linked activities (a scheduling activity network) is scheduled together by a single call to a method of an ABAP-OO class. A complete scheduling result is thus obtained for all activities. The network of activities that is coherently processed can be a sub-network of the scheduling activity network or the complete scheduling activity network.*

***Time unit***

*is needed to define dates and durations. Each duration needs a number and a time unit. A time unit is optional to describe a date with a certain precision. For example a date with the unit 'day' will not be precise at 'seconds' level, it will be rounded to a precision of a day. A date without a time unit is a simple time stamp. This is also called time unit 'exact'. A date with a time unit is a time slice with a start and end time stamp.*

***Time unit assignments***

*Defines for a set of date types a time unit that will be used in the scheduling algorithm. Time unit assignment can be re-used in several different business processes. Date types that are used in the business process but that have no time unit assigned are treated as dates with time unit 'exact'. This means that they are simple time stamps only.*

***Scheduling calendar***

*An SAP factory calendar or an SAP time stream.*

***Business process modeler (BPM)***

*An application of SAP Basis that will be developed within Basis release 6.20/6.30. The BPM will be an enhancement of the already-existing workflow builder. Using the BPM, a business process can be defined by setting up a network of process steps.*

***Entry point date***

*The date that is passed to SCM scheduling as the starting point for the scheduling algorithm. The date has to exist in the timetable (complete or derived) of the activity network. SCM scheduling may change this date; it is not fixed in the scheduling algorithm.*

***Alias of a business process***

*Refers to an existing workflow definition via a mapping table. The scheduling can thus be called using an alias of a business process instead of a workflow ID.*

## 1.1 Obsolete Glossary

After the design review, the glossary was partially changed (after discussion with the translator/info developer). Note that only the glossary in this document has been adapted; the terms used in the text of this document remain unchanged. This section provides the relations between terms used in the text (obsolete glossary) and the revised glossary terms.

***Extended timetable format***

is now called *complete timetable*

***Condensed timetable format***

is now called *derived timetable*

***Scheduling scheme***

is now called *scheduling schema*

***Duration type***

is now called *activity type category*

***Atomic pieces of the scheduling***

are now called *building blocks*

***External processing***

is now called *delegation*

***1st grade external processing***

is now called *partial delegation*

***2nd grade external processing***

is now called *complete delegation*

***Coherent external processing***

is now called *coherent complete delegation*

***Precision profile***

is now called *time unit assignments*

***Standard process***

is now called *business process with alias*

# 2 Rough Design

**Note that to understand this design document, you should read the glossary section first before studying the document in detail.**

## 2.1 Introduction

SCM scheduling is intended for use in the new SCM application 'Fulfillment Coordination' (SCFC) which deals with the definition, execution and control of business processes, focusing on the logistic part of the processes. SCFC will be able to deal with a variety of different logistic processes that can be freely defined by means of a BPM. The processes will be executed in an open environment of several applications, components and systems. SCFC will make use of

services provided by other applications in order to execute the several steps of a business process.

Scheduling of the business processes is needed, probably at several stages of the execution process. When a business process is begun, a preview is needed in order to estimate the timetable that describes the dates when certain steps of the business process will (have to) be executed. This happens in the same way as the transportation and shipment scheduling that can be performed when a CRM (R/3) sales order is created and a confirmed delivery date is determined. In addition, the execution of a process may differ from this preview (due to exceptions), which will lead to the need for re-scheduling of the process in order to get an accurately adjusted timetable in fulfillment coordination.

Fulfillment coordination works in a heterogeneous environment of several components/systems by making use of services provided by other applications. Fulfillment coordination itself will provide a generic framework that integrates these applications into an overall business process.

SCM scheduling will provide a generic framework for scheduling pre-defined business processes in an environment of several components/systems. SCM scheduling will make use of a process definition created by a BPM and a set of meta-data, and the scheduling scheme, which is used to define the determination of several procedures that are performed within the scheduling. For example, the meta data of the scheme may define an access sequence to several services which may be used to provide master data needed for the scheduling (duration, calendar, time zone).

The framework of SCM scheduling technically makes use of dynamic calls to ABAP-OO classes that implement certain interfaces defined by SCM scheduling. This technique is the same as the BADI technique. The core framework of SCM scheduling, which is discussed in this document, does not contain those ABAP-OO classes that may be used within the definition of the scheduling scheme. The standard case will be that implementation of those classes and definition/shipping of schemes that make use of them has to be done by applications that make use of SCM scheduling (such as fulfillment coordination). Nevertheless, extension of the core framework of SCM scheduling may be done later on in order to create ABAP-OO classes that are shipped by the package of SCM scheduling and can be used as elements of the definition of application-specific schemes. The creation of those ABAP-OO classes owned by SCM scheduling can be done when the master data available for SCM 4.0 is clear. This may include classes that access a master data manager (MDM) or that access a business rule repository that holds the information on scheduling master data.

Given the fact that the development presented here is a generic framework, it may be used by several other applications in addition to fulfillment coordination. The SCM scheduling framework will be compatible (to a large extent) with SAP R/3 SD delivery and transportation scheduling, to SAP APO transportation and shipment scheduling and to the ABAP based (infinite) scheduling of SAP APO planning applications (such as SNP).

## 2.2 Architecture and Behavior

### 2.2.1 Description of a Scheduling Activity Network

SCM scheduling is able to schedule a network of activities. The definition of this network consists of a description of a single activity and a description of the network.

**A single activity consists of the following:**

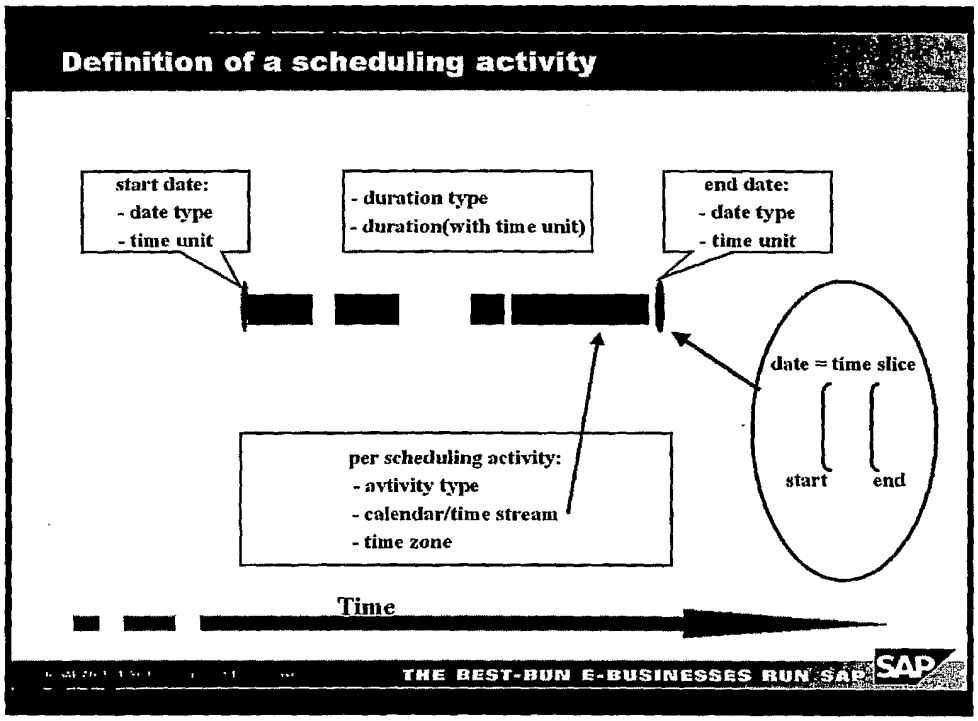
- Activity type
- Date type of the start of the activity
- Date type of the end of the activity

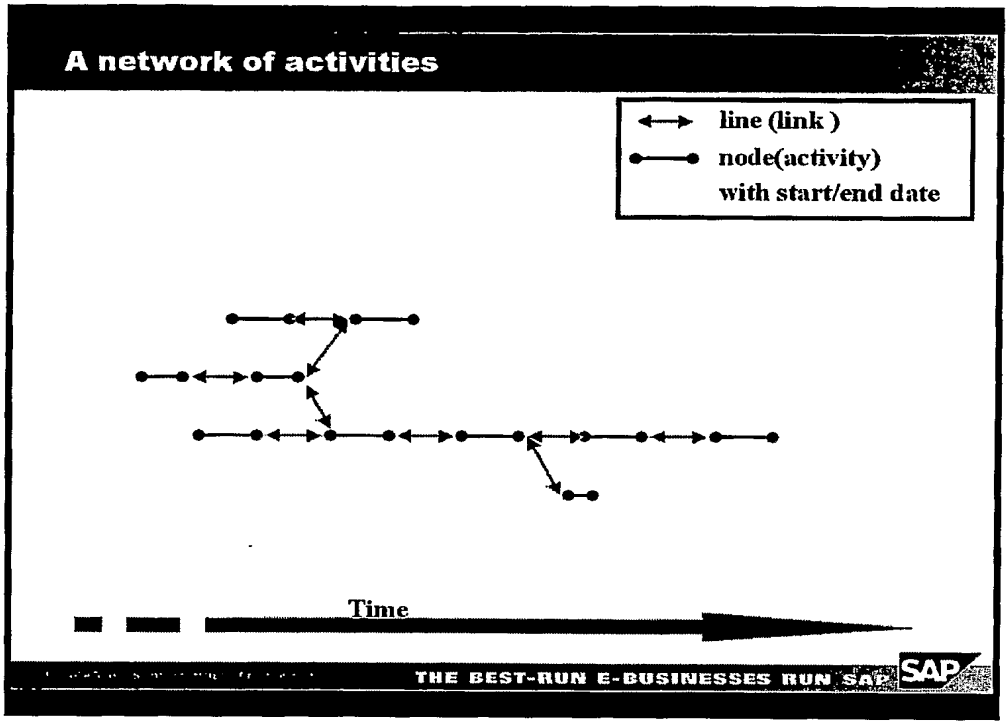
- Duration type of the activity
- Calendar of the activity
- Time zone of the activity
- Time unit of the start date, time unit of the duration and time unit of the end date

**An activity network consists of:**

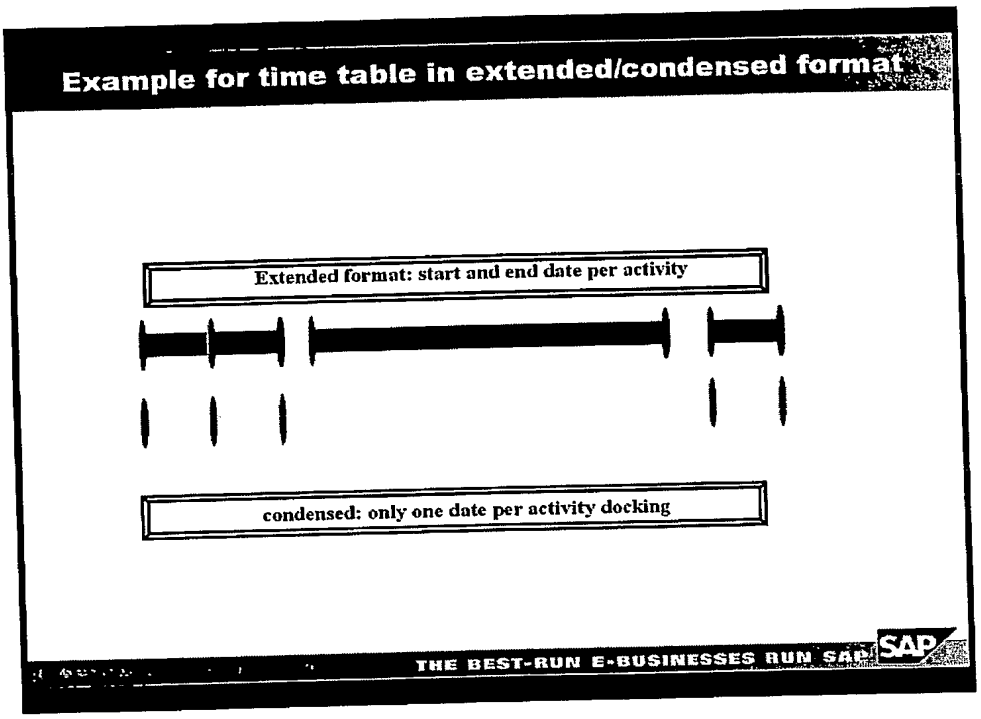
- Nodes (equivalent to a scheduling activity)
- Lines (link predecessor and successor nodes)

The following figures show the description of a scheduling activity and an activity network.






SCM scheduling can describe a network of activities in time by means of two different timetable formats. The extended timetable format consists of two dates for each activity (start and end-date of the activity). The condensed timetable format consists of a list of dates that can be derived from the extended timetable. The following figure shows an example of a simple activity graph with an extended and a condensed timetable.




## 2.2.2 SCM Scheduling Framework Data Models and Data Buffers

The scheduling needs a data model to define the static part of a scheme and to define the dynamic part of a scheme. In addition to the scheme definition managed by SCM scheduling an activity network definition will be maintained using the BPM (workflow builder). Thus, the activity network definition will be managed by the workflow data model and is not part of the SCM scheduling framework.

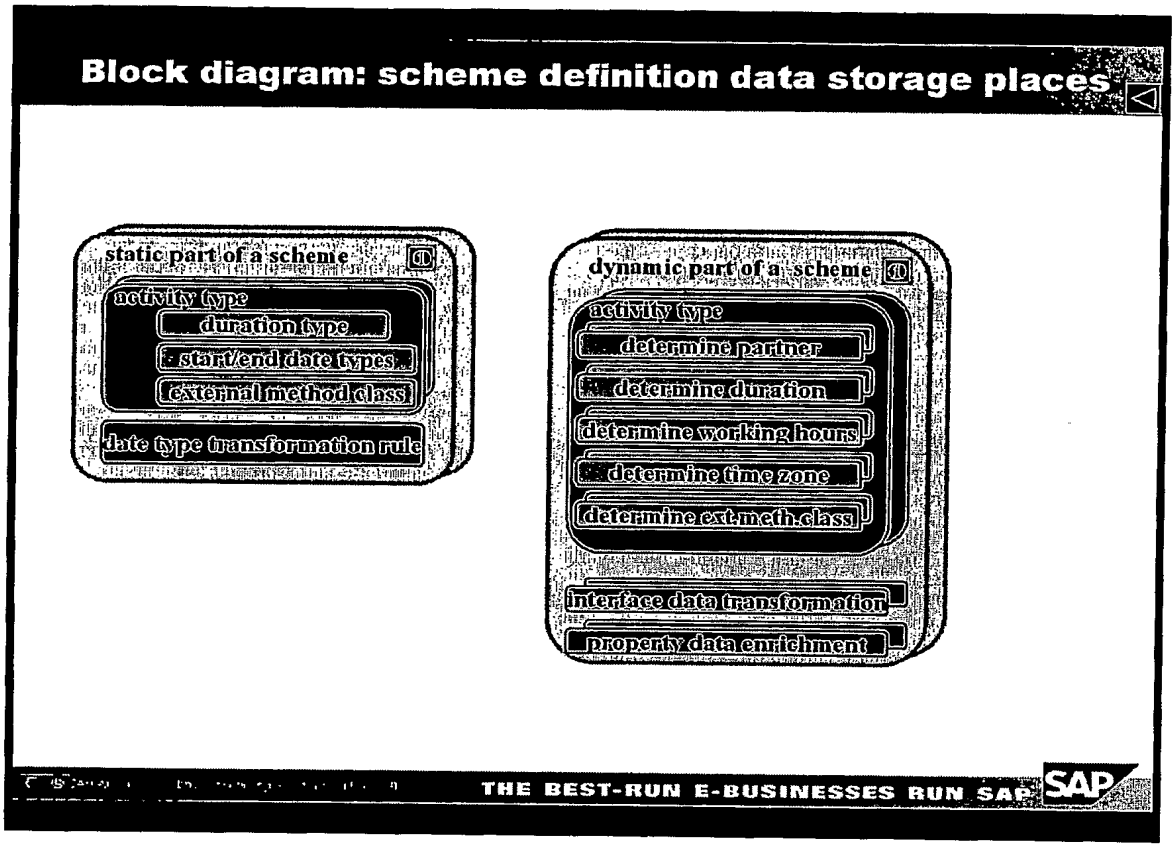
SCM scheduling will comprise master data to define weight and volume groups.

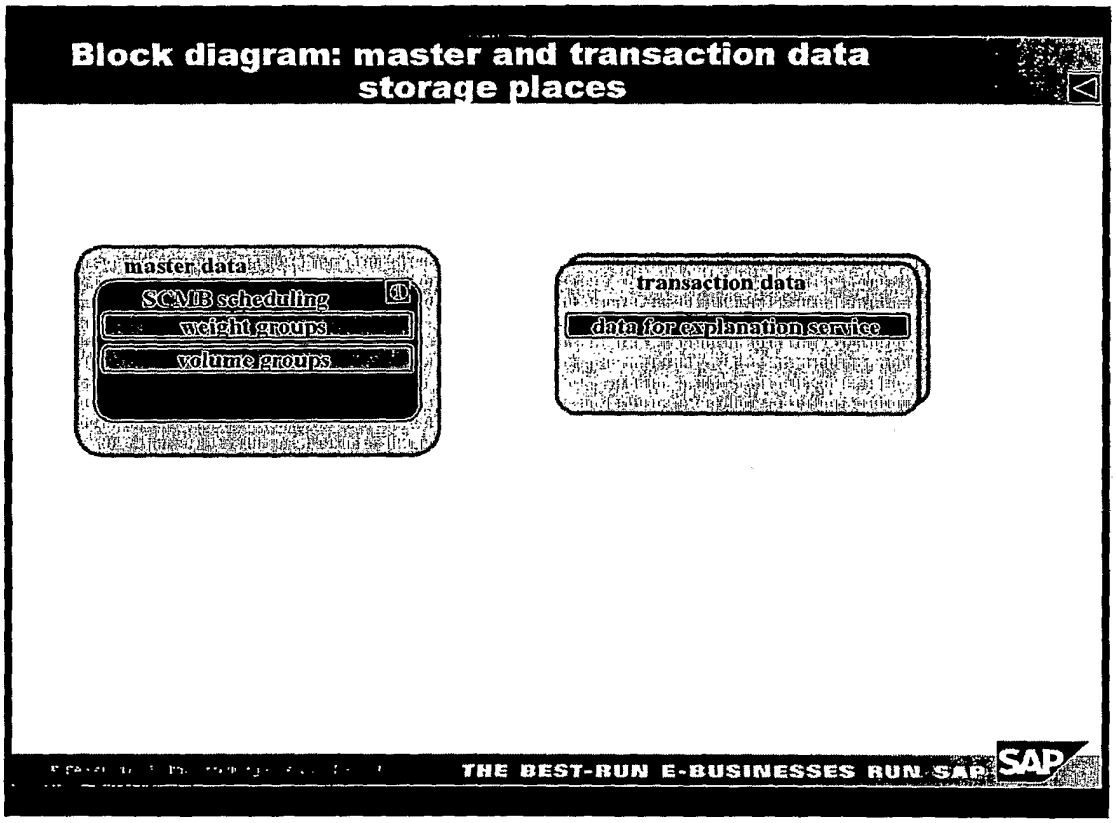
In addition to SCM scheduling master data, classes that are set up in the dynamic part of the scheme (and that implement the interfaces specified by SCM scheduling) may use any additional master data available to the components to which these implementation classes belong.

Amongst others, master data that is planned to be available via a master data interface layer of SCM Basis might be used. Another example would be the usage of business rule repository (BRR) conditions/rules. These condition records may be evaluated within the determiner methods of the dynamic scheme definition. The BRR belongs to SAP Basis and is thus available for use inside SCM scheduling determiner procedures.

In addition to master data, SCM scheduling has to manage transaction data. This includes the data that is needed for the explanation service of SCM scheduling. This explanation data consists of a data buffer (an object in the memory) and persistent data (by means of database tables). An explanation of previously processed scheduling calls is thus made possible within a transactional context and also offline (without the transactional context in which the scheduling call was processed).

The following figures show in a block diagram an overview of the data storage places of the scheduling. Please note the special syntax of this kind of block diagrams as defined in SAPNET.


Block diagram: scheme definition data storage places
static part of a scheme
activity type
duration type
start/end date types
external method class
date type transformation rule
dynamic part of a scheme
activity type
determine partner
determine duration
determine working hours
determine time zone
determine ext.meth.class
interface data transformation
property data enrichment
THE BEST-RUN E-BUSINESSES RUN SAP
SAP




### 2.2.3 SCM scheduling Services

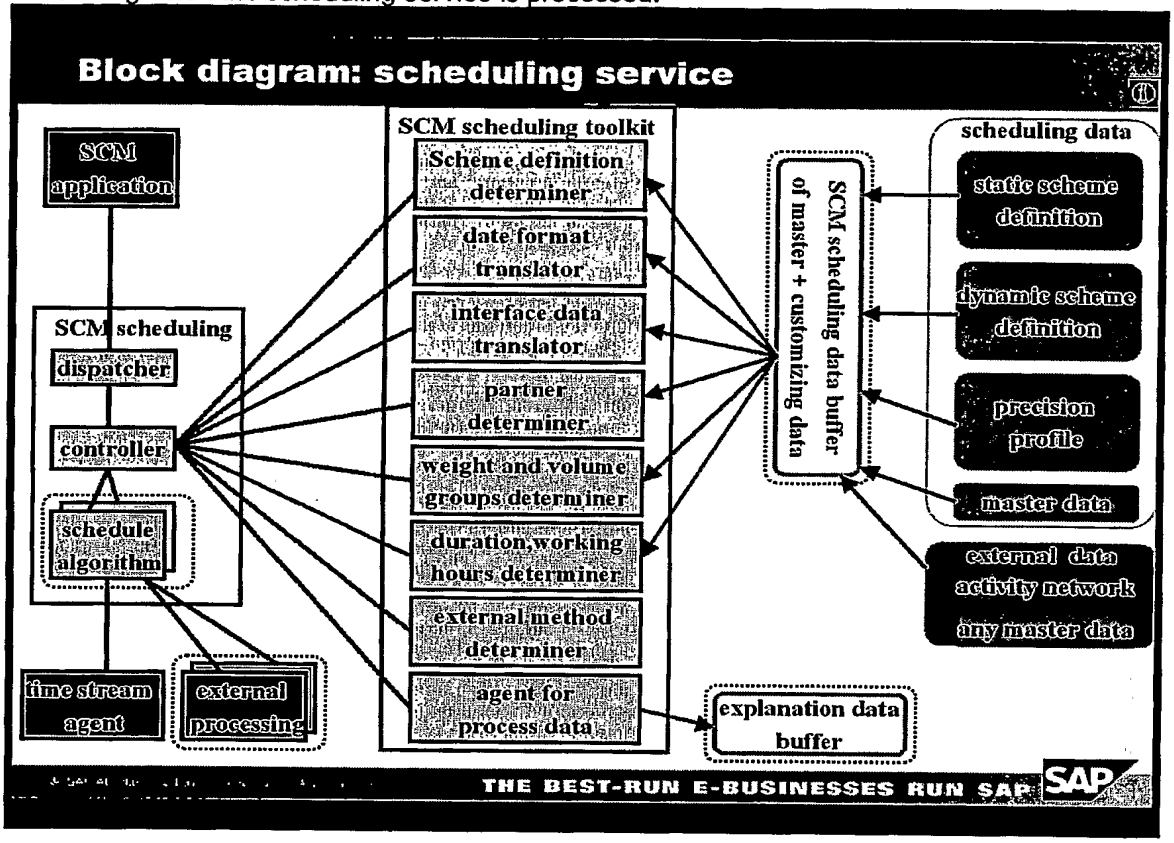
SCM scheduling comprises services for the following:

- Scheduling a business process
- Presenting and explaining the scheduling result
- Posting SCM scheduling transaction data and external processing

Note that the block diagrams in the following sections are in accordance with the graphic and syntax standards as described in SAPNet. The wording used in the text is also agrees with this syntax. Do not misunderstand the word 'agent'. An 'agent' in this context is a piece of a functional program, such as an ABAP-OO class method or a function module, in contrast to other types of building blocks such as data storage places.

#### 2.2.3.1 Overview: Processing of the Scheduling Service

The following figure shows, in a block diagram, an overview of the active components of the scheduling when the scheduling service is processed:




An external application calls the scheduling service via its API. The SCM scheduling first routes all methods of the API through an entry agent which then carries out a functional dispatch. This entry agent provides a single point of entry to the SCM scheduling framework internally although the framework provides several API methods.

If scheduling is requested, the entry agent will dispatch to the scheduling controller. The controller first performs a pre-step that makes use of the scheduling toolkit in order to derive the static scheme and the dynamic scheme. This results in a description of the complete graph of activities that includes information from the scheme, the activity network and the master data. This graph of activities description holds all information necessary for a subsequent scheduling engine. The controller then invokes the scheduling engine.

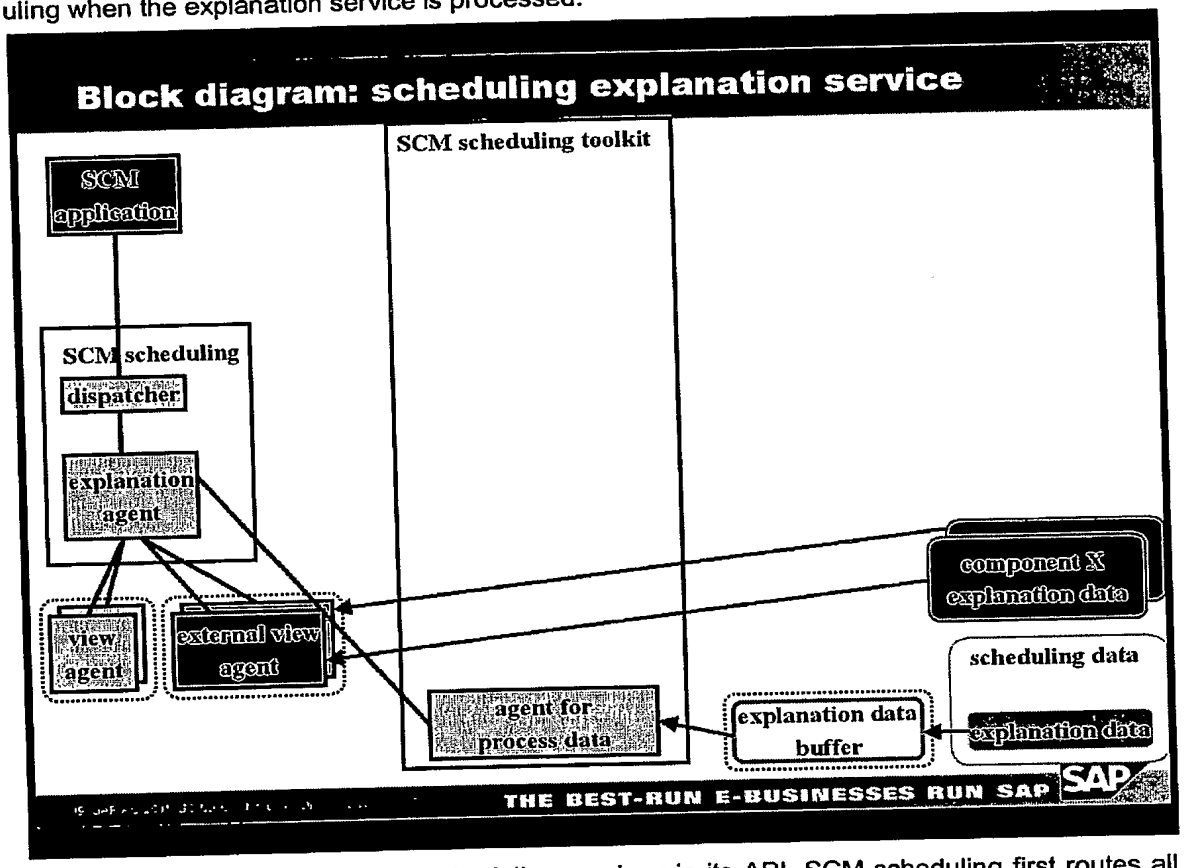
The scheduling engine performs the pure scheduling algorithm for a network of activities.

Scheduling of individual activities is done by the activity objects that are instances of an ABAP-OO class. Activity objects are contained in the description of the graph of activities built by the controller. They make use of an interface of a time stream agent that is able to carry out basic scheduling functions by means of SAP Basis package SZTI. Package SZTI provides the function needed to calculate with different time units taking time zones, factory calendars and time streams into consideration. Activity objects also make use of an interface for external processing, which might have to be called for certain activities in order to set start- or end-date of the activity, or in order to schedule the complete activity using an external method.

After the scheduling engine has completed its work, the controller stores the result and additional explanation data in the explanation data buffer.

### 2.2.3.2 Overview: Processing of the Explanation Service

The following figure shows, in a block diagram, an overview of active components of the scheduling when the explanation service is processed:

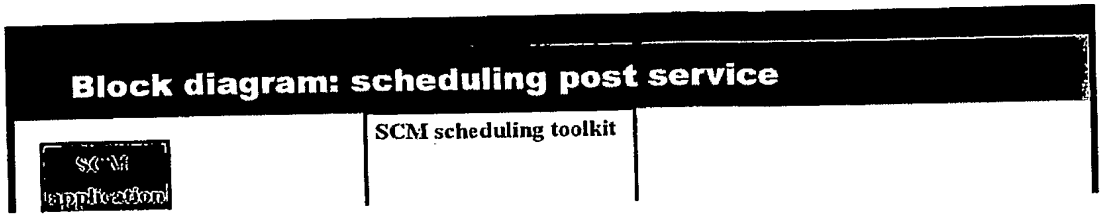



An external application calls the scheduling service via its API. SCM scheduling first routes all methods of the API through an entry agent which then carries out a functional dispatch. If an explanation is requested, the entry agent dispatcher will call the explanation agent.

The explanation agent first retrieves data of a preceding scheduling service call (which may still be stored in the explanation data buffer or which is available via database storage) and then provides several views. The explanation agent will contain a view controller in order to manage the different views. Views to explain external processing will also be needed.

### 2.2.3.3 Overview: Processing of the Scheduling Post Service

The following figure shows, in a block diagram, an overview of active components of the scheduling when the post service is processed:



An external application calls the scheduling service via its API. SCM scheduling first routes all methods of the API through an entry agent which then carries out a functional dispatch.

If a post is requested, the entry agent dispatcher will call the post agent. The post agent may retrieve the explanation data buffer and post this data to the database.

The post agent will raise a public event 'POST' of class /SCMB/CL_SC_CONT. This enables all instances of classes that were previously used for external processing to execute their specific postings (data saving). These classes will not be destroyed when the scheduling method is carried out if they subscribe to this event.

## 2.3 Program Interfaces/Integration

### 2.3.1 Software Component, Application Component, Package Structure and Namespace

Implementation will be carried out using the ABAP package concept and a namespace.

SCM scheduling will consist of one package /SCMB/SC01 that will contain the objects and have a package interface. The objects will use namespace /SCMB/ (SCM Basis). Package /SCMB/SC01 belongs to application component SCM-BAS-SCH (SCM Basis) with transport layer SCMB (Supply Chain Management Basis). (Note that application component SCM-BAS-SCH has to be created. As of yet, only SCM-BAS exists). The application component SCM-BAS-SCH belongs to software component SAP_SCMB (Supply Chain Basis). The software component describes the packaging of software in respect to software delivery. Software component SAP_SCMB is defined by SCM Basis structure package /SCMB/SC_STRUCTURE. The scheduling package /SCMB/SC01 is contained in this structure package.

Scheduling package /SCMB/SC01 has only BASIS and ABA interfaces as use accesses as maintained in the package definition. Thus, scheduling package /SCMB/SC01 can make use of BASIS, ABA and SCMB objects only. SCM scheduling will not use objects of any other packages of SCMB. Thus a unique structure package for SCM scheduling (and therefore a "common ABAP software part") will be possible. This would enable the delivery of SCM scheduling independently of SCM Basis.

Any package (not belonging to SCM Basis) that has declared a use access of SCMB can make use of SCM scheduling without generating an error in the package check.

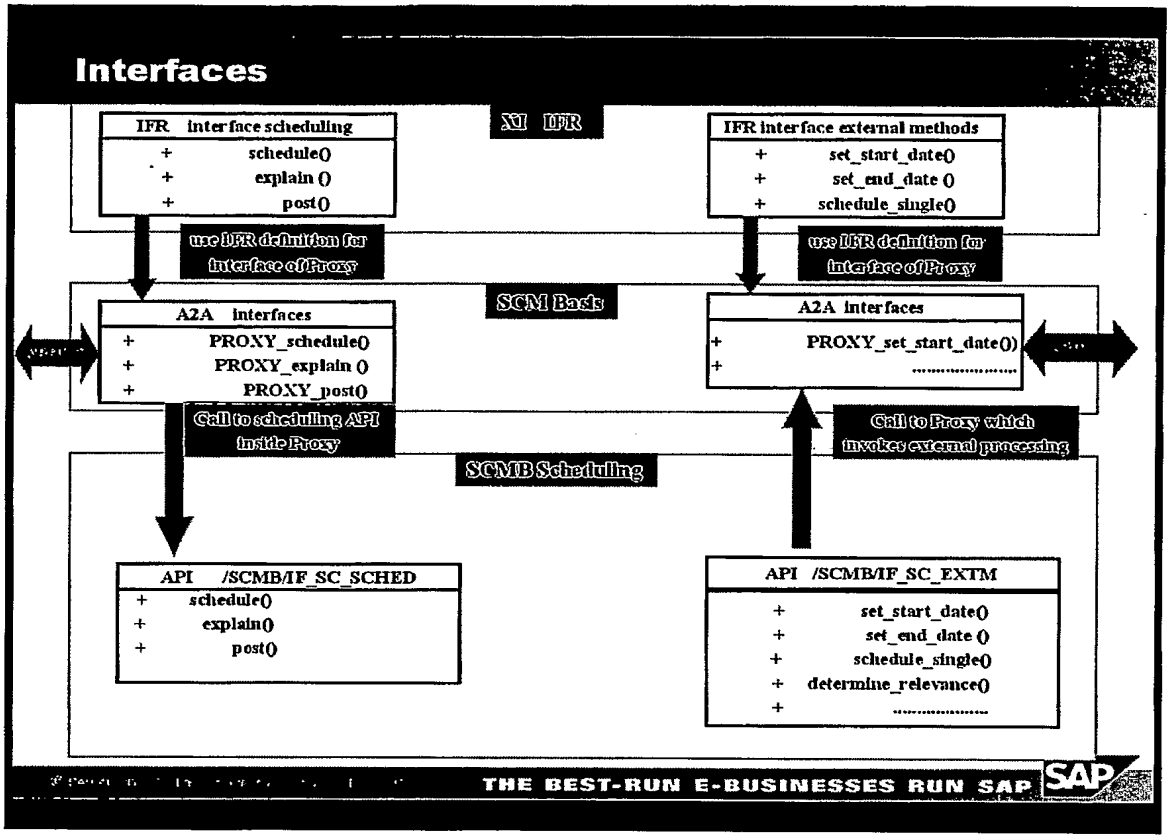
Note that we intend to place SCM scheduling in the SCM Basis layer (as described here). The Development Coordination Meeting (DCM) still has to decide on this.

### 2.3.2 Interfaces

SCM Scheduling will contain several interfaces. The most important ones are:

- An API for local scheduling (no remote access). This is the ABAP-OO interface /SCMB/IF_SC_SCHED that contains static methods for scheduling, explanation, and posting.
- A function group of RFC modules that correspond to the methods of the API. These RFC modules may be used for communication within a landscape of several SAP systems (like CRM-APO-R/3). The RFC modules will be in accordance with the SAP BAPI standard. (A BAPI will not be created at first, because this kind of communication will be carried out via the XI infrastructure).
- An IFR definition analogous to the API interface will be created. Based on this IFR interface, applications that make use of SCM scheduling may create an A2A interface that uses the XI infrastructure. The IFR interface has to be defined in such a way that in the PROXY-functions, which are generated by the A2A interfaces, the ABAP-OO API can be implemented. According to the current status of discussions, the PROXY functions generated for the A2A interfaces of all applications belonging to SCM will be created as part of SCM Basis.
- In addition to the scheduling API, an API definition for so-called external processing will be created. With this API (and possible proxies, function- or BAPI calls running within the implementation of this API) the scheduling processing can be carried out (in part) by other applications/components/systems/partners.

The following figure gives an overview of the architecture of A2A interfaces that might make use of the XI infrastructure. Application 'APPL X' may use a proxy as an A2A interface to perform an SCM scheduling service call. The proxy may be generated using an IFR definition of the interface and by using the SCM scheduling ABAP-OO interface /SCMB/IF_SC_SCHED to implement the proxy. Within the scheduling, external processing may be performed using the ABAP-OO interface /SCMB/IF_SC_EXTM. Implementation of the interface may use a proxy for performing the processing by means of another application 'APPL Y'.




## 2.4 User Interfaces

SCM Scheduling comprises the following user interfaces:

- Maintenance of 'atomic' objects.
- Maintenance of scheme (static and dynamic part)
- Maintenance of precision profile
- Maintenance of activity network
- Test and simulation report
- Explanation service

For the maintenance of 'atomic' objects, the static part of a scheme, the dynamic part of a scheme and the precision profile standard view maintenance will be used.

The activity network will be maintained using the BPM (workflow builder). By setting a property for a workflow-step this step will be flagged as a scheduling activity. In the BPM the scheme and

the precision profile that should be used to schedule the process will also be maintained as a property at header level.

The test and simulation report will enable you to define all the input data of the scheduling API as report input parameters. The input data can be provided on the entry screen of the report and, if requested, in an additional dialog box to define a table of properties. The report output will be the explanation service of the scheduling.

The explanation service will enable you to show several views. The standard views of SCM scheduling will be based on ALV listings and graphic packages of SAP Basis technology. Later on, the explanation service may be upgraded to Web-Dynpro technology or BSP technology.

# 3 Detailed Design

## 3.1 Definition of the Scheduling Scheme

A scheduling scheme is a set of meta data that describes a framework that is used for the technical processing of the scheduling. It is a definition on how to handle individual activities of business processes within the scheduling and it is a definition on how to determine the master data needed for scheduling, such as duration and calendar of an activity.

The data model of the scheduling scheme consists of the definition of 'atomic' objects (which are the components of a scheme) and the definition of the scheme itself. 'Atomic objects' may be reused in several different schemes. 'Atomic objects' are:

- Activity type (like 'Load at issuing plant' or 'Load ship at harbor', for example): Defines the activities that can be used to define scheduling schemes. For each activity an ABAP-OO class can be maintained as default at the activity level for external processing. Activity types are defined by tables /SCMB/TSCACTI and /SCMB/TSCACTIT (see the figure below).
- Duration type (like 'loading', 'packing', 'picking'): Several activities of a scheduling scheme may use the same duration type. For example 'loading' might be used for an activity at the issuing plant but may be used in addition for an activity at a load transfer point (at a harbor ...). Based on the duration type conditions/rules for determination of the duration of an activity may be setup. Duration types are defined by tables /SCMB/TSCDURA and /SCMB/TSCDURAT (see the figure below).
- Date type (like 'delivery date', 'material availability date'): Date types are used to describe the start and end dates of an activity. This is the so-called extended timetable format. Date types may also be used to define the condensed timetable format. The condensed format contains a subset of all start/end dates of activities of the process. A date type may be used for the extended and for the condensed timetable format at the same time. The date type does not carry a category that differentiates between the extended/condensed timetable formats. Date types are defined by tables /SCMB/TSCDATE and /SCMB/TSCDATET (see the figure below).

Each of the 'atomic objects' has a technical name and a language-dependent text that can be used for a UI presentation. To maintain this, data views will be defined. Standard view maintenance will be generated for these views.

A scheduling scheme comprises a list of activity types for which the framework definition has been set up. Only activity types that belong to the same scheme can be used in the definition of a business process (one process definition of the BPM can refer to a single scheme only). A scheme definition can be reused in several different business process definitions, with the restriction, that the business processes use only those activity types that belong to the scheme.

Scheduling schemes have to be pre-defined during a setup phase when the business processes, that is, activity network, are defined. Applications that make use of the scheduling service in order to schedule individual business processes pass the key of the activity network definition (that is, workflow-ID) to the scheduling. This activity network defines which activities are present in the business process and how they are related in time. In addition, the activity network definition contains a reference to the scheme that will be used in the scheduling.

The 'atomic objects' and the scheme will have a protected SAP namespace by using table TRESC. This allows the delivery of standard business scenarios by delivering table datasets that describe standard atomic objects and standard schemes. Therefore all tables will have delivery class 'G' (Customizing table, protected against SAP update, only INS allowed). Note that the namespace definition of table TRESC is the major reason for the existence of the sym-

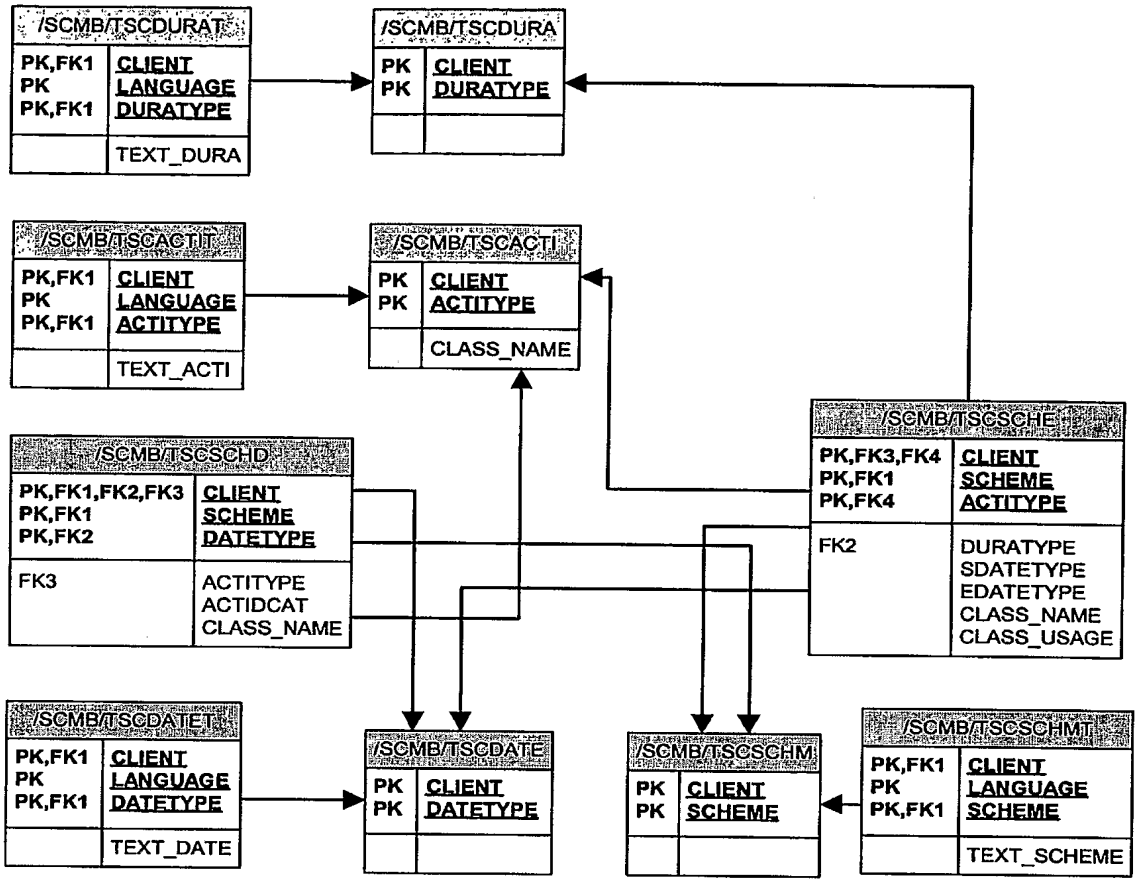
bolic names for ABAP-OO classes that will be used in the SCM scheduling framework (see section 3.1.2).

A scheduling scheme consists of a static part of the scheme and a dynamic part of a scheme.

### 3.1.1 Definition of the Static Part of a Scheduling Scheme

Definition of the static part of scheme:

- Table /SCMB/TSCSCHM defines the scheme identifiers.
- Table /SCMB/TSCSCHMT provides a language-dependent description text for each scheme identifier.
- A scheme consists of a list of activity types (see table /SCMB/TSCSCHE)
- For each activity type of a scheme the duration type, date type of start and end of the activity has to be defined. This is done by the fields DURATYPE, SDATETYPE and EDATETYPE of table /SCMB/TSCSCHE. Note the restriction that the date types used for start/end of the activities have to be unique within a scheme. This allows for unambiguous transformation between date types and start/end-points of activities. The uniqueness of the date types will be guaranteed by the secondary indexes of table /SCMB/TSCSCHE and by checking the input data in the view maintenance (event 05).
- For each activity of a scheme an ABAP-OO class for external processing can be set as default at the level scheme/activity (field CLASS_NAME in table /SCMB/TSCSCHE). The ABAP-OO class is set by a symbolic name CLASS_NAME. The symbolic names CLASS_NAME are defined in the dynamic part of the scheme using table /SCMB/TSCCLASS.
- For each activity of a scheme a control parameter that enables external processing can be maintained (field CLASS_USAGE of table /SCMB/TSCSCHE). This parameter can have the values: No external method class use, always take default from activity definition, always take default from definition of static scheme, always take dynamic determination depending on application data, evaluation by priority: Dynamic determination, static scheme definition, activity definition
- For each activity of a scheme the start and end can be mapped to a date type that is used for the condensed timetable format. This mapping is done using table /SCMB/TSCSCHD. Field DATETYPE denotes a date type of the condensed timetable format. ACTITYPE denotes an activity type of the scheme and ACTIDCAT denotes whether the start or end of the activity should be mapped (ACTIDCAT may have the values A ='start of activities', B ='end of activity'). In addition, a symbolic class name may be defined by field CLASS_NAME for a more complex transformation algorithm between the two timetable formats of a scheme. Note that an unambiguous transformation between a date of the condensed format and a date of the extended format must be possible in both directions of transformation.




### 3.1.2 Definition of the Dynamic Part of a Scheme

The dynamic part of a scheme defines the behavior of the scheduling for a certain scheme depending on application data that is passed to the scheduling service call. The definition of the dynamic part comprises:

- A definition for ABAP-OO classes (via table /SCMB/TSCCLASS) that can be used in the scheme definition. This definition indicates for which purpose the class may be used. This is indicated by the fields USE_* which are simple flags. The definition is done by first defining a symbolic name for an ABAP-OO class (field CLASS_NAME) and by linking an ABAP-OO class to this symbolic name (field SEOCLSNAME). It is thus possible to ship the datasets of table /SCMB/TSCCLASS. Note that the table has delivery category 'G' and a defined namespace for the symbolic class names. In addition, using table /SCMB/TSCCLASST, a text describing the functional role of the class can be defined for each symbolic class name.
- A definition of the transformation of externally provided properties (passed to the scheduling via an interface) into the internally used table of properties. This is done using table /SCMB/TSCIMAP that might have several datasets for an internal property DATA_NAME. Mapping might be carried out directly by passing the value of a property PROP_NAME (provided by the calling application via the interface of the scheduling) or it might be performed in a more complex way using an ABAP-OO class that implements the mapping inter-

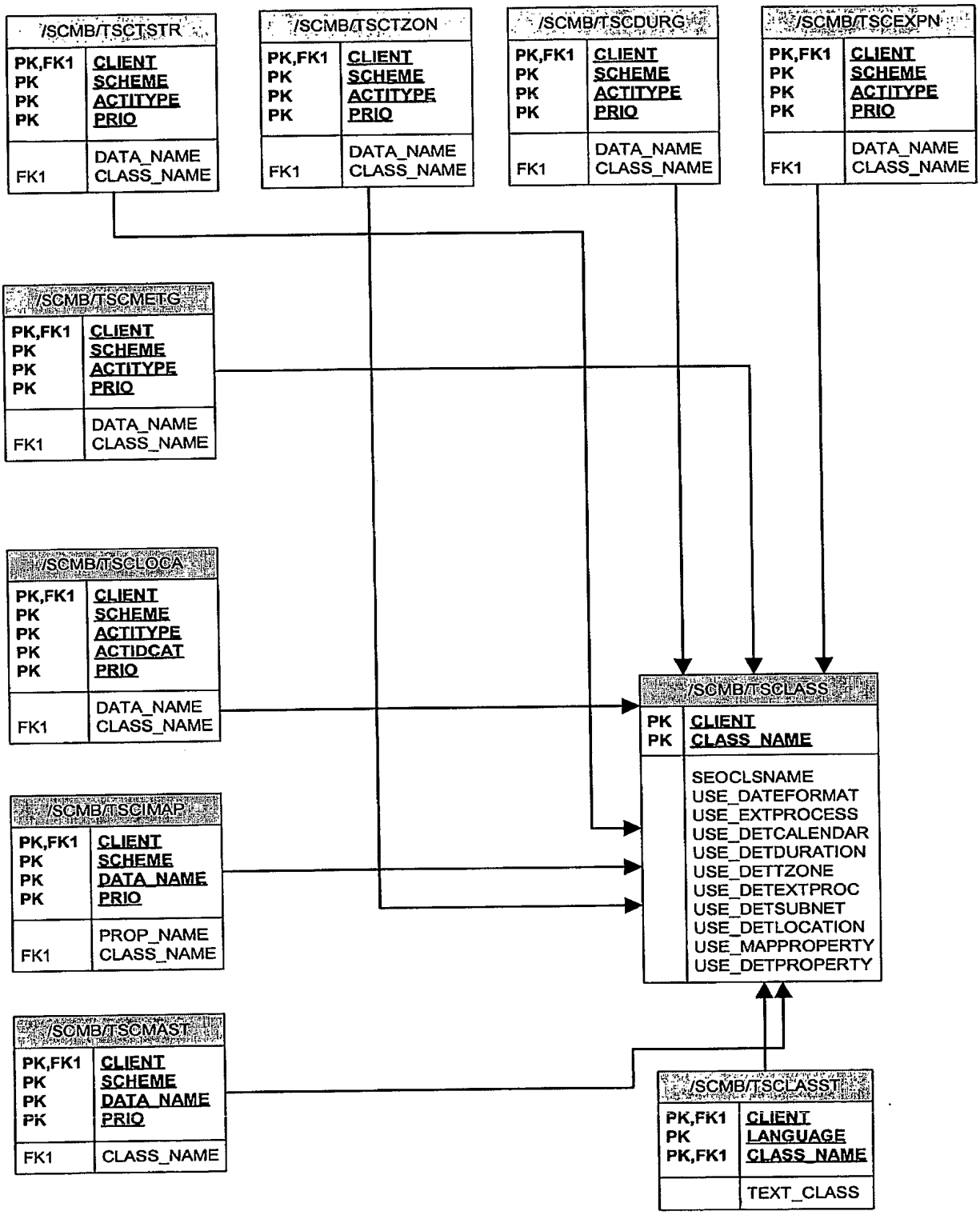
face /SCMB/IF_SC_IMAP. In addition to the mapping, all properties that are passed via SCM scheduling interface are transferred with a name identical to an internally used property. This is only true for those externally provided property names that are not used as DATA_NAME in table /SCMB/TSCIMAP. The internally used table of properties is the starting point for the determination procedures described below.

- A definition of location/business partner determination. This is done using table /SCMB/TSCLOCA. For each start/end date of an activity of a scheme a location/partner may be determined directly by value mapping from an internal property DATA_NAME or by using an ABAP-OO class that implements interface /SCMB/IF_SC_LOCA. These locations/partner IDs may subsequently be used to determine the duration, working hours and time zone of the activity. Within the scheduling framework several different types of business partners/locations may be used. For example one may make use of the ABA business partner with ABAP data element BU_PARTNER (CHAR10) or the APO location with ABAP data element /SAPAPO/LOCID (CHAR22). Inside the scheduling framework the partner will be treated in a generic manner as a property of the activity. Applications that make use of SCM scheduling are responsible for the consistency of the partner determination setup and the procedures setup (see below) that may make use of these partner IDs. The procedures that use the partner IDs have to be adopted to the type of partner.
- A definition of enrichment of internally used property data. This is done using table /SCMB/TSCMAST. For each scheme a set of internal property names (field DATA_NAME) can be defined. For each property name several ABAP-OO classes, which implement interface /SCMB/IF_SC_MAST, can be specified with different priorities. In case the property is not yet known in the scheduling (because it could not be determined by transformation from the properties provided via the scheduling interface), SCM scheduling will use the ABAP-OO classes in order to determine the property. This is the how master data or application-specific data might be read during SCM scheduling. Access to the data is not performed in the core coding of the SCM scheduling framework but is done within the dynamically chosen ABAP-OO classes. These implementation classes may belong to a package/software component different to SCM scheduling.
- A definition of the working hour (time stream) determination. This is done using table /SCMB/TSCTSTR. For each activity of a scheme the time stream id might be determined by directly passing the value of an internal property or it might be determined by means of an ABAP-OO class that implements interface /SCMB/IF_SC_TSTR. In order to schedule with the time unit 'day' (see section 3.7) in addition to a time stream, a factory calendar may be determined by means of the ABAP-OO class. The value of a property may consist of a time stream ID concatenated by a factory calendar ID. A factory calendar can also be determined by means of a property. For an activity of a scheme several datasets of table /SCMB/TSCTSTR might exist with different priorities.
- A definition of duration determination. This is done using table /SCMB/TSCDURG and is analogous to time stream determination. Note that duration consists of a value and a unit (DDIC: TIMEDURA and TIMEUNITNA). The value of the internal property that is used for duration determination has to consist of a value concatenated with a unit (DDIC: TIMEDURA followed by TIMEUNITNA).
- A definition of time zone determination of activities. This is done using table /SCMB/TSCTZON and analogous to time stream determination. The time zone of activities is needed to round dates and to schedule in case a factory calendar is used. Note that each activity has a single time zone. The time zones used in the scheduling algorithm to round start and end dates are identical to the time zone used to schedule the duration with regards to a calendar. If an application wants to make use of different time zones, it has to model an activity network consisting of several activities. For example a transport from Germany to US-East that will include the two different time zones has to be modeled by three activities: Goods issue, transport and goods receipt. Goods issue and goods receipt

will have the time zones CET and EST, whereas the transport activity will have the time zone that corresponds to the working calendar of the ship.

- A definition of the determination of ABAP-OO classes for external processing. This is done by table /SCMB/TSCMETG and is analogous to time stream determination. This determination may take place depending on the CLASS_USAGE field of table /SCMB/TSCSCHE.
- A definition on how to evaluate if sub-networks, which are attached to an activity, will be expanded in the scheduling. This is done by table /SCMB/TSCEXPN (see 3.2.2 for details).

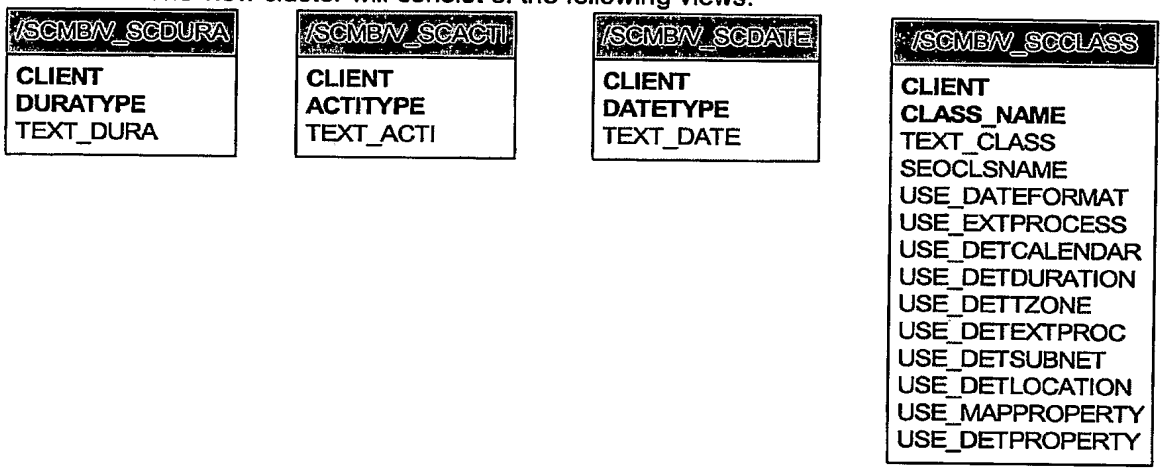
Note that the four identical-looking tables /SCMB/TSCTSTR, /SCMB/TSCTZON, /SCMB/TSCDURG, /SCMB/TSCEXPN will differ by the ABAP data elements used for the field CLASS_NAME. Thus, dedicated search help and documentation can be assigned to each use case of the classes.


/SCMB/TSCTSTR
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCTZON
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCDURG
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCEXPN
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCMETG
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCLOCA
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK ACTIDCAT
PK PRIO
FK1 DATA_NAME CLASS_NAME
/SCMB/TSCMAP
PK,FK1 CLIENT
PK SCHEME
PK DATA_NAME
PK PRIO
FK1 PROP_NAME CLASS_NAME
/SCMB/TSCMAST
PK,FK1 CLIENT
PK SCHEME
PK DATA_NAME
PK PRIO
FK1 CLASS_NAME
/SCMB/TSCLASS
PK CLIENT
PK CLASS_NAME
SEOCLSNAME
USE_DATEFORMAT
USE_EXTPROCESS
USE_DETCALENDAR
USE_DETDURATION
USE_DETTZONE
USE_DETEXTPROC
USE_DETSUBNET
USE_DETLOCATION
USE_MAPPROPERTY
USE_DETPROPERTY
/SCMB/TSCLASST
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 CLASS_NAME
TEXT_CLASS

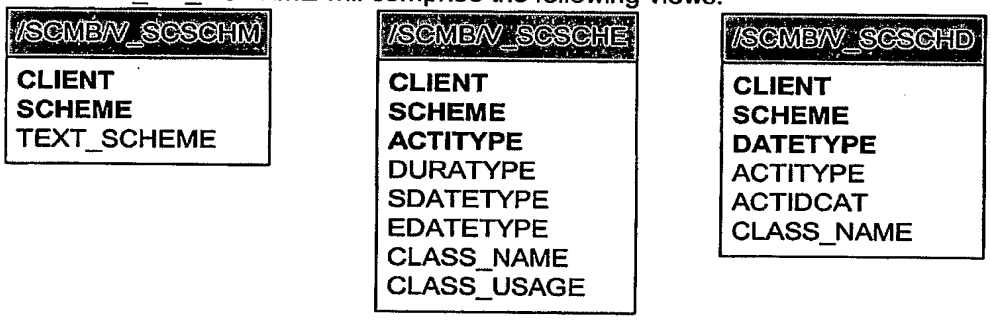
### 3.1.3 View Cluster and Views for Maintenance of the Scheme

To maintain the static and dynamic part of the scheduling scheme, standard generated view maintenance will be used.

To maintain the atomic objects and table /SCMB/TSCCLASS (which describes the set of available ABAP-OO classes), a view cluster /SCMB/VC_SC_ATOMIC will be used. By means of this view cluster all these database tables will be maintained within a single user dialog. Transaction /SCMB/SC_SETUP_A which makes use of SM34 and view cluster /SCMB/VC_SC_ATOMIC will be created. The view cluster will consist of the following views:




To maintain the static part of the scheme view cluster /SCMB/VC_SC_SCHEME and transaction code /SCMB/SC_SETUP_S will be used in the same way. This view cluster will be structured so that the data is maintained depending on the scheme. View cluster /SCMB/VC_SC_SCHEME will comprise the following views:




In addition, view cluster /SCMB/VC_SC_SCHEME will comprise a view for each table of the dynamic part of the scheme. These are:

- /SCMB/V_SCTSTR
- /SCMB/V_SCTZON
- /SCMB/V_SCDURG
- /SCMB/V_SCEXPN
- /SCMB/V_SCMETG

- /SCMB/V_SCLOCA
- /SCMB/V_SCIMAP
- /SCMB/V_SCMAST

## 3.2 Definition of the Activity Network

With the definition of the activity network a business process is defined from point of view of the scheduling. The activity network defines the set of activities present in a business process as well as their relation in time. The network definition has to include a cross-reference to a scheme that will be used during the scheduling of the activity net.

The network definition will **not** be carried out by SCM scheduling tools. The network definition will be carried out by means of the BPM (workflow builder). SCM scheduling will use API function module SWD_API_WORKFLOW_GRAPHIC_GET to read the network. As of Basis release 6.20 SP4 the interface of this API will contain all data needed for the function described below. A prototype (see reference in section **Fehler! Verweisquelle konnte nicht gefunden werden.Fehler! Verweisquelle konnte nicht gefunden werden.Fehler! Verweisquelle konnte nicht gefunden werden.**), which used an internal workflow function `SWD_WF_DEFINITION_READ`, already showed the feasibility of the integration of BPM and SCM scheduling described above.

The network definition used internally by the scheduling comprises a table of activities and a table of lines that denote predecessor – successor connections. This network definition is derived from the workflow data read by the workflow API.

In the BPM in the basic data section, which is the data at header level of a workflow, a property that denotes the key of a scheme has to be defined. The name of the property will be a fixed name 'SAP.SCM.BAS.SCH.SCHEME'. With this property the scheme that will be used in the scheduling of the process is denoted.

In the step maintenance of the BPM, you can define that a property that denotes the technical name of the business process step as a scheduling activity. The name of the property will be a fixed name 'SAP.SCM.BAS.SCH.ACTITYPE'. If this property is present for a workflow step, this step is relevant for scheduling. If this property is not present for a workflow step the step is not relevant for scheduling. The value of the property denotes the activity type of the workflow step.

A network of scheduling activities can be derived from the BPM network of steps. This network of scheduling activities may not contain all BPM steps due to neglecting steps that are irrelevant to scheduling. The set of lines of the network of scheduling activities is derived from the lines of the BPM network of steps. BPM lines that end at a step irrelevant to scheduling are concatenated. These concatenated lines are the lines of the activity network that have a step relevant to scheduling at both ends of the line.

Note that in the creation of the network of scheduling activities only the step property relevant/irrelevant for scheduling is used. The workflow node type is not taken into account. This node type indicates the type of step (for example, activity, condition, multiple condition, while, until, fork...). Thus all steps that are not real activities (like a condition) will not be relevant for scheduling. Furthermore, these types of steps have to be used carefully in the BPM definition, because they are not present and/or are used with different semantics in the scheduling activity network. For example a condition step will lead to a network of scheduling activities with two lines (a line for the true result and a line for the false result of the condition) because the condition itself is not evaluated in the determination of the activity network. Thus, a condition is used in the same way as a fork.

The BPM (workflow builder) allows for top-down modeling of a business process that results in steps that contain a sub-network (sub flow). The header property '...SCHEME' of sub-networks has to be identical to the scheme used for the overall network definition (parent workflow). SCM

scheduling will be able to expand these sub-networks in order to obtain the complete activity network. Steps that contain sub-networks do not need to have the property '...ACTITYPE'. In the latter case, these steps themselves will never be part of the scheduling network (but their sub-network will be part of the scheduling network).

Note that BPM (workflow) process definitions can be shipped. This allows the definition of standard processes by SAP development.

### 3.2.1 Process Determination and Standard Processes

Applications that use SCM scheduling can provide a workflow ID at the interface of the scheduling service. In this case, the activity network of this workflow ID will be used by the scheduling service. This modus is intended for applications that make use of configurable processes. These workflows have to be set up by those applications meaning that the definition data of these workflows may be shipped by those applications.

SCM scheduling will define and ship some standard process definitions on its own. Each of these standard processes will have an alias. Mapping between workflow ID and alias will be carried out using a table with delivery class 'G' (Customizing table, protected against SAP update, only INS allowed). If an application calls the scheduling service and does not pass on a workflow ID, a given alias will be used to determine a standard process. For example, one standard process will be named 'SCHEDL' and will be a process definition the same as the fixed process definition used in SAP APO transportation and shipment scheduling. By means of these standard processes, SCM scheduling can perform a scheduling with an activity net that is consistent to existing SAP scheduling functions such as SAP R/3 and SAP APO shipment and transportation scheduling and SAP APO SNP scheduling.

SCM scheduling will not internally perform a process re-determination in case of changed properties due to external processing (see section 3.8.11). A process re-determination might be needed, for example, if a sourcing service such as Global ATP is used for external processing. It will be up to the calling application to re-determine the process and call SCM scheduling again with a new process description.

### 3.2.2 Zooming Into Details of an Activity Network: Expand a Sub-Network

If the activity network definition contains process steps that are sub-networks, two different types of process steps have to be handled differently in the scheduling:

1) A process step contains a sub-network and has the property '...ACTITYPE'. In this case two possibilities exist:

a) Expanding the sub-network during scheduling

b) Do not expand the sub-network but take the parent step as a single scheduling activity as an estimate of the sub-network instead

2) A step that contains a sub-network does not have the property '...ACTITYPE'. In this case the sub-network always has to be expanded in the scheduling since no representation of this part of the process would be there in the scheduling otherwise.

During the scheduling of a network, a decision has to be made for steps of type 1) whether the sub-network should be expanded or not. This will be done according to the settings of table /SCMB/TSCEXPN in the same way as the other tables of the dynamic part of the scheme (like table /SCMB/TSCMETG or /SCMB/TSCTSTR). Thus the expansion of individual activities can be controlled by several different procedures (via interface data and via different ABAP-OO class implementations of an interface) with different priorities. For example, you may set up a dataset in table /SCMB/TSCEXPN for each activity with DATA_NAME ='*ACTITYPE*_EXPAND' where *ACTITYPE* is the name of the activity as given in the BPM step which holds the sub-network. Using interface data (table of properties) of the scheduling, the calling application may

then control the expansion of certain sub-networks by submitting the relevant property '*ACTITYPE*_EXPAND' with value 'TRUE'.

Zooming into the details of a business process and thus scheduling it by taking into account more details may be used to have different views of a business process. This could be depending on the current process step where the business process is. You might have different levels of zoom for a preview of a business process, and during the different stages when the process is partially evolved.

Note that the use of sub-networks to zoom into details of a process is limited to those sub-networks that were defined during the design phase of a business process.

### 3.2.3 Determination of Sub-Network for Coherent External Processing

External processing will be possible not only for single activities but for a sub-network that comprises several activities contained in the overall network. This will be possible for activities that are directly linked together, that perform 2nd grade external processing (that is, the complete activity is determined by external processing) and all activities make use of the same ABAP-OO implementation class for external processing.

Because these ABAP-OO classes are determined dynamically during runtime, a static definition of the sub-networks that can be processed coherently (together) is not possible. The determination of sub-networks for coherent external processing will be done during runtime according to the criteria given above.

The interface for external processing includes a relevance interface method that is used by the scheduling during the pre-step of the controller. This relevance interface allows the scheduling to check whether 2nd grade external processing will be performed. In addition, the scheduling can check via this interface whether sub-network processing is allowed and if it can be handled by this ABAP-OO implementation class. If this is the case, activities that fulfill the criteria of a sub-network will be scheduled coherently by a single call to the external processing ABAP-OO class that schedules the complete sub-network.

Note that these sub-networks may have several inbound lines and several outbound lines. They are not limited to a trivial sub-network consisting of single in- and outbound line.

Note also that the use of sub-networks for coherent external processing is not driven by the design phase of the process (it is not due to network master data). It is due to the settings in the scheme master data that are evaluated during runtime and which describe the technical settings for processing. From a business point of view, the packaging for coherent external processing is just a technical detail; the business process itself remains unchanged.

The determination of sub-networks for coherent external processing is done in the pre-step of the controller. Thus the schedule algorithm makes use of the network definition and does not perform any network determinations itself. The overall architecture thus remains open for other input sources of the network definition.

## 3.3 Dynamic Use of ABAP-OO Classes

Given the above data design, SCM scheduling will make use of ABAP-OO classes that implement interfaces that are defined by the SCM scheduling package. This may take place in external processing as well as in the determination of data needed for the scheduling.

Technically, the mechanism is similar to the BADI technique: An interface is defined and implementing ABAP-OO classes supply the functional part. As in the BADI technique, interface definition belongs to the application that makes use of interface methods (SCM scheduling) whereas the implementing ABAP-OO classes may belong to any other software component available on the same system.

SCM scheduling will use instances of these classes, but only with a narrowing type cast. This will ensure that SCM scheduling will use the generic interface that is defined by the scheduling only. The following short test coding shows the possibility of creating an ABAP code that serves these requirements:

```
REPORT ZADTEST_OO.

interface I1.
    methods testm.
endinterface.

class C1 definition.
  public section.
    interfaces I1.
endclass.
class C1 implementation.
  method I1~testm.
    write: 'test ok'.
  endmethod.
endclass.

start-of-selection.
   data: refi1 type ref to I1.

create object refi1 type ('C1').
 call method refi1->testm.
```

Interface I1 is an example of an interface that will be defined by SCM scheduling. Class C1 is an example of those classes that might be defined and implemented by other applications. Using the data design given above, it is possible to dynamically determine the class and therefore a specific implementation.

Due to the fact that the class type of the instance is used only dynamically in the code and the whole static code makes use of the SCM interface only, this is compatible with the ABAP package check. This is true as long as the package to which the class belongs is allowed to use the interface definition of SCM scheduling. This is the case because all SCM applications are allowed to use any SCM Basis object. From the point of view of software packaging, the technique used here is analogous to the BADI technique.

In addition to the possibility of using ABAP-OO classes belonging to other packages/software components (a kind of direct use of application API's) you may hide the interface between SCM scheduling and the applications using ABAP-OO classes that are placed in the SCM scheduling package and that perform a call to other applications by calling BAPI's or PROXY functions of

the XI infrastructure. It is also possible to include cross-system processing in the internal processing of the implementing ABAP-OO classes. The technique used for cross-system processing (RFC, BAPI or XI) is irrelevant for the SCM scheduling framework because it is hidden behind the scheduling API for external processing.

Note that external processing is limited to synchronous processing. SCM scheduling will not enable asynchronous step-by-step scheduling of a process. Given that this may become important when external processing is done via the XI infrastructure by external partners, asynchronous external processing will be a matter of a future release of SCM scheduling. The need for possible asynchronous external processing will be reflected in the design structure by capturing the complete data needed for scheduling the network in one memory object. This object would be the context that would have to be saved temporarily between several asynchronous external-processing steps and restored when going on after receiving the result of an asynchronous external processing. A prerequisite of this future function will be the 'Scheduling a Package of Requests With External Processing' as described in section 3.8.12.1.

### 3.3.1 Search Help and Validity Check for ABAP-OO Classes Implementing Certain Interfaces

A search help for allowed ABAP-OO classes (classes which implement a certain interface) can easily be created. This has already been shown in the prototype implementation. Thus the usability data maintenance that defines a scheme can be improved. (Beyond the prototype example that defines a search help by direct use of a DDIC view several other examples such as function SWD_HELP_F4_CLASSES_BY_IF exist).

In addition, the standard view maintenance provides an exit-point (event 05) were a validity check can be implemented. This validity check can be created analogous to the search help.

In addition to the validity check during scheme maintenance, a validity check at runtime is needed, because of the possibility of dynamically determining an ABAP-OO class by means of a method. If an ABAP-OO class, which will be used in the scheduling, is not available in the system where the SCM scheduling is processing, an exception will be created. In order to check the consistency of the scheme definitions (static and dynamic part) present in a system, a report that can validate that all ABAP-OO classes used in scheme definitions are available on the system will be created

### 3.3.2 Generic Data Transfer to External Processing

ABAP-OO classes that determine duration, calendars and so on, work with the table of internal properties and the locations/partners of the activities as input data only. This data is also included in the interface for external processing. Given the fact that external processing may be much more complex than the determination of master data, the interface of external processing will additionally include a generic data container of type ref to IF_SWF_CNT_CONTAINER. This is the container as used by the BPM and BRR (workflow). This container may be passed to SCM scheduling by applications. SCM scheduling will pass the container to each external processing. In contrast to the table of properties, the container allows complex data types to be passed.

If Fulfillment Coordination makes use of SCM scheduling, this container may be identical to the process context container used for starting a workflow process.

### 3.3.3 Preventing Recursive Calls Within External Processing

SCM scheduling may be used within external processing methods. This may lead to recursive SCM scheduling and external processing calls.

An example would be:

Application *FC* calls SCM scheduling. The scheduling invokes external processing for the start date of activity *PICK* by means of method/class *XCLASS*. The external processing may call the scheduling. This recursive scheduling will carry out external processing again for start date of activity *PICK* by means of method/class *XCLASS*. *And so on...*

The recursive call can be prevented if the scheduling carried out within the external processing is prohibited from calling the same external processing again. The SCM scheduling interface will have an optional input table that can be used to prohibit external processing. Each line of the table holds an activity type and flags that describe which external processing will not be carried out (1st grade start/end date or 2nd grade). The table can be used to prohibit external processing for several activities, which is useful in case of multiple recursive calls to SCM scheduling. If external processing is called by SCM scheduling, this table will be given to the interface of external processing so that it can be passed to the SCM scheduling when it is called internally. The table will at least contain the data that describes the external processing that is called at this processing step. It may contain more data in the case of multiple recursive calls.

It is up to the outer ABAP-OO class that performs the external processing to decide if recursive external processing will not be carried out. The decision on this will be complex because cases exist when recursive scheduling, which makes use of the same external classes, makes sense. SCM scheduling will monitor which external processings are still running by means of a static table in class /SCMB/CL_SC_CALC. If recursive calls are forbidden and the static table indicates that an external processing is still running, SCM scheduling will raise an exception when the same external processing is to be called.

## 3.4 Functional ABAP-OO Class Structure

The rough design described in section 3 will be implemented by means of ABAP-OO. This means that all functional parts (agents) shown as rectangles in the block diagrams will be implemented as ABAP-OO classes. In addition, ABAP-OO classes will be needed to implement data storage (which is not database tables but some kind of data buffers in the memory). The management of SCM scheduling database tables (scheme definition and process data) will also be encapsulated in ABAP-OO classes and/or function groups of generated view maintenance.

## 3.5 Scheduling Master Data

SCM scheduling will enable you to maintain weight and volume groups. These groups might be subsequently used as parameters for duration and time stream determination.

| /SCMB/TSCWGRP | |
|---|---|
| PK | CLIENT |
| PK | WGROUP |
| | WMAX |

| /SCMB/TSCVGRP | |
|---|---|
| PK | CLIENT |
| PK | VGROUP |
| | VMAX |

All other master data needed for scheduling will not be stored in tables belonging to the core framework of SCM scheduling. Thus the core framework of SCM scheduling will not provide any tools for maintaining this data. Master data of other components used by SCM scheduling may comprise

- SAP Basis: Time streams and factory calendars

- SAP Basis: Business rule repository which may hold conditions to determine duration and time streams/factory calendars
- SCM Basis: An interface for master data retrieval of location and product
- Any other master data of other components.

Given that SCM scheduling enables you to configure the determination procedures by setting up ABAP-OO classes in the dynamic part of the scheme, those classes (which implement the interfaces of SCM scheduling) may use any other application's/component's/system's data.

In general, SCM scheduling provides a framework for the configuration of determination procedures of master data but does not have its own master data. The only exceptions are master data of weight and volume groups that are identical to SAP R/3 and SAP APO volume/weight group master data.

### 3.5.1 Time Zone Interpretation of Time Stream and Factory Calendar Master Data

SAP Basis time streams (table TTSTR) and SAP Basis factory calendars are used as master data to describe working hours/days. Depending on the time unit used for scheduling, either a time stream or a factory calendar is used (see section 3.7).

A prerequisite of time streams that will be used by SCM scheduling is that they are stored in a database with time zone UTC. This is the case if field TTSTR-TZONE is either initial or equal to 'UTC'. This method of storing time stream data is already used by SAP R/3, SAP APO and SAP CRM. For example, if you define a time stream using transaction /SAPAPO/CALENDAR with time zone CET, in SAP APO all period data defined in this transaction will be converted to time zone UTC before storing it into the time stream database. A drawback of using of time streams with time zone UTC is that for several time zones with identical working hours (in local time zone), a single time stream does not suffice. For each time zone an individual time stream has to be created as master data instead. Due to backward compatibility reasons, SCM scheduling will make use of time streams stored in time zone UTC only. Thus SCM scheduling may directly use SAP Basis time streams that are defined by master data maintenance tools of existing SAP applications (like /SAPAPO/CALENDAR).

In the case of factory calendars, a time zone cannot be specified in the master data itself. Thus a factory calendar is a generic definition of working days independent of a time zone. A factory calendar that is assigned to an activity will be used in the local time zone of the activity.

## 3.6 Use of (Application-Specific) Master Data in the Scheduling

In general, all master data that will be used by the scheduling may be passed to the scheduling via the table of properties of API interface method /SCMB/IF_SC_SCHED->SCHEDULE. This master data may be used to determine scheduling-specific data (such as the determination of the duration of an activity by means of conditions) and may also be used for external processing.

In addition, data enrichment can be performed within the scheduling service by adding additional property datasets. These properties are then available during the subsequent determination of scheduling specific data (such as the duration of an activity).

Data enrichment is carried out at the beginning of the pre step of the controller according to the settings of table /SCMB/TSCMAST. In this table you assign several names of ABAP-OO classes with different priorities to the internal name of a property. By means of an ABAP-OO class the value of an internal property may be derived by any access to master data, if the value of the property is not yet known due to properties given via the interface to the scheduling service.

The settings of /SCMB/TSCMAST are scheme-dependent. A scheme can thus be used to restrict possible master data read accesses to a certain number of ABAP-OO classes. For example, in the case of a scheme that does not contain a transportation activity the scheduling does not need detailed knowledge on transportation properties such as transportation zones of delivery plant and goods recipient. In this case, there is no need for a dataset in /SCMB/TSCMAST that ensures that the transportation zone is known within the scheduling (due to interface data or due to reading master data).

## 3.7 Scheduling With Time Units

SCM scheduling will make use of package SZTI that was originally developed for CRM but was moved to Basis layer with Basis release 6.10. This package contains the interface IF_TIMEUNIT_CALC and implementing classes for time units second, minute, hour, day, week, month, quarter, year and exact (which means that no time unit is given). All units (except unit *exact*) describe dates by means of a time slice with a start and an end timestamp. In the case of the unit 'exact', the start and end timestamps of the time slices are identical.

The interface IF_TIMEUNIT_CALC contains the methods round and move (scheduling) that allow for rounding and scheduling with a given time unit. The scheduling with units exact, second, minute and hour may consider a time stream. The scheduling with unit day may consider a factory calendar. Scheduling with larger units does not consider any factory calendar or time stream.

Both methods, round and move (schedule), may consider a time zone. In the case of rounding, this time zone is used for the conversion of a UTC time stamp to a local date and time that is used in the rounding mechanism. The rounding mechanism always rounds a given time stamp to a time stamp that is equal to the start of a time slice. This start of a time slice is called a *rounded timestamp*. The rounding is done in the frame of local time zone. For example in the case of unit 'day', the rounded timestamp will be equivalent to the start of a day in the local time zone. In case of scheduling with time unit 'day' with method 'move', the time zone is needed for conversion to the local date because the scheduling has to be executed in the local frame in order to take a factory calendar into account correctly.

Package SZTI comprises database table TIMECUUT and TIMECUUTT that hold the information of available time units and their implementing classes. These datasets will be shipped by SCM scheduling. SCM scheduling will ship datasets equal to those that are shipped by CRM. In the long run these datasets will be shipped with SAP Basis layer. For the time unit 'exact' no dataset has to be shipped by table TIMECUUT. An instance for processing with this time unit is always created by the framework of package SZTI (CL_TIMEUNIT_BROKER).

SCM scheduling will use the time unit of the duration, time zone and calendar of the activity (which are determined according to the dynamic part of the scheduling scheme by means of tables /SCMB/TSCDURG, /SCMB/TSCTZON and /SCMB/TSCTSTR) to perform method 'move' which is the basic scheduling of a duration taking into account a factory calendar or a time stream.

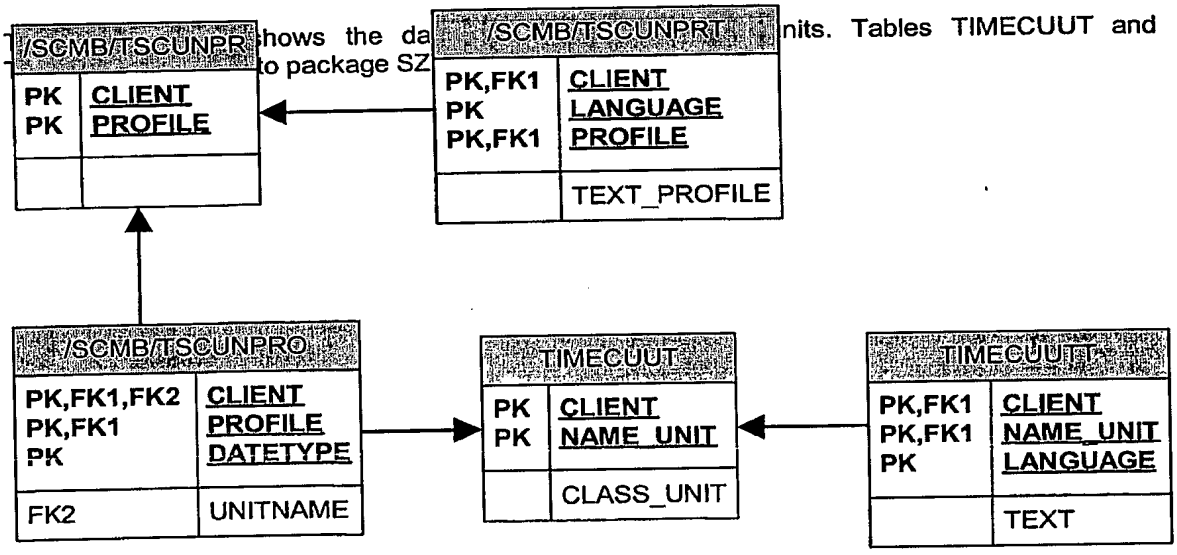
A framework for the definition of the time units of the start- and end dates of the activities has to be set up within SCM scheduling. This definition is carried out by the SCM scheduling precision profile. Each profile comprises a list of date types with a unit assigned to it. The precision profile might be given in the header properties of the BPM by property name SAP.SCM-BAS-SCH.UNITPROF or can be specified call by call via the interface of the SCM scheduling methods. Date types that are not contained in the precision profile but are used for scheduling will be taken into account with time unit 'exact'.

The time units of the dates will be used to round the start and end dates of an activity. A single activity may comprise three different time units for start date, end date and duration. Time units

of start and end date are provided by the precision profile, the time unit of the duration is provided by the procedures that are set up in the dynamic part of the scheme.

The result of the scheduling will consist of complex date objects. These result date objects of the scheduling will consist not only of the rounded single timestamp but will also include a time slice of type CL_TIMESLICE (this class is contained in SAP Basis package SZTI). The time slice of type CL_TIMESLICE denotes start and end time stamp of the time slice in UTC. In addition, time unit and time zone information will be included in each date object. The time zone is needed for correct interpretation of dates with time unit not equal to 'exact' because the rounding is done in the frame of local time zone. Thus the slices make sense only in the local time zone of the date.

The input interface of the scheduling will consist of the same complex date objects. Entry point dates and fixed dates have to be specified using these complex date objects. In the case of the input interface, only the start of time slice of type CL_TIMESLICE is obligatory. If time zone and time unit are not given in the interface data, the dates will be rounded using time zone and time unit determined within the scheduling before using the input dates within the scheduling algorithm. If time zone and time unit are provided in the interface data, the dates will first be rounded using these data before they are used in the scheduling algorithm.



The time units are extensible, because the tables TIMECUUT and TIMECUUTT have delivery category 'E'. Maintenance of time units can be done using Basis view V_TIMECUUT.

To maintain the SCM scheduling unit profile, a view cluster that comprises view /SCMB/V_SCUNPRT and view /SCMB/V_SCUNPRO will be used. To maintain this view, cluster transaction /SCMB/SC_UNPROF will be created.

## 3.8 Scheduling Algorithm in Detail

**The following features will be accounted for in the algorithm:**

1) Scheduling a network of activities including time units and external processing of activities or sub-networks.
2) Take a list of additional entry point dates ('Roundtrip emulation') into account. (See SCM scheduling specification document for details)
3) Take constraint on earliest date (which may be specified individually for each date type) into account
4) Push or pull optimization (see SCM scheduling specification document for details)
5) Take date fixing (fixing indicator and date values are provided by caller) into account.
6) Splitting of the scheduling result due to external processing that may have several results.
7) Convergence strategy (maximum number of iterations)

**The following features will not be supported by the scheduling algorithm:**

1) Allowed time interval for entry point. The algorithm does not internally perform a series of scheduling procedures that start with different entry point dates out of the allowed time interval and thus does not determine the best result out of the various scheduling procedures. This kind of logic has to be done by the calling application. Nevertheless, a time interval for the entry point date is used implicitly due to the time unit mechanism. An entry point time stamp (in UTC) is used in the scheduling algorithm after rounding the entry date using the time unit mechanism.

### 3.8.1 Scheduling of a Single Activity

This makes use of the following:

- Entry point date *ENTRY (which* is a time stamp in UTC)
- Duration *DURA* with time unit *DUNIT* of the activity
- Time zone *TZONE* of the activity
- Factory calendar/time stream *CAL* of the activity
- Time unit *SUNIT* and *EUNIT* of start and end date of the activity

The sequence algorithm for a forward scheduling will be as follows:

1) Take entry point time stamp ENTRY and validate it against the calendar of the activity. This is done using if_timeunit_calc->move with TZONE, DURA = 0, SUNIT and CAL. The result is a time stamp that is an active working time/day according to the time stream/factory calendar assigned to unit *SUNIT.*
2) Schedule result of 1) using if_timeunit_calc->move with *TZONE, DURA, DUNIT* and *CAL.* The result is a time stamp that is an active working time/day according to the time stream/factory calendar assigned to unit *DUNIT.*

3) Take result of step 2) and validate it against the calendar of the activity. This is done using if_timeunit_calc->move with TZONE, DURA = 0, EUNIT and CAL. The result is a time stamp that is an active working time/day according to the time stream/factory calendar assigned to unit *EUNIT*.

4) Round result of 1) using if_timeunit_calc->round with *TZONE* and *SUNIT*. The result is the time slice that contains the result of step 1).

5) Round result of 3) using if_timeunit_calc->round with *TZONE* and *EUNIT*. The result is the time slice that contains the result of step 3).

Results of step 4) and 5) are the start and end dates of the activity as determined by the scheduling algorithm. These result dates are complex date objects (that include a time slice).

Note that due to the rounding of steps 4) and 5) the rounded timestamp of the end date of the activity (this is the start time stamp of the end date time slice) is not always later than the rounded time stamp of the start date of the activity. If the start time unit is hour, the end time unit is day and the rounding is always done to the beginning of a time slice, the rounded timestamp of the start of the activity may be later than the rounded time stamp of the end of the activity. Due to the fact that the result does not consist of the rounded time stamp only but comprises a time slice with start/end time stamp (and a time zone) this feature of the rounded time stamps is not an error. The rounded time stamps alone are not meaningful!

Note that the time slices of the result date objects contain at least one working hour second, but they may not completely consist of working hour seconds. This is due to steps 4) and 5) that round a working hour timestamp to a time slice. The rounding mechanism does not take a time stream/factory calendar into account.

Note that the precision assigned to the duration of an activity will not be better than the precision that is used for the dates. For example, use of *DUNIT* = day (that causes the use of a factory calendar in step 2) of the above algorithm) together with a precision of hour for start- and end date works without problems.

For example, use of *DUNIT* = second (that causes the use of a time stream in step 2) of the above algorithm) together with a precision of day for start- and end date may result in rounded time stamps that are not a valid working hour second according to the time stream. In this example, a time stream that describes working hours with the precision of a second is spread out by the rounding mechanisms. Nevertheless, the latter example is allowed. It can be useful, for example, to schedule activities with a working duration of several days, taking working hours into account and rounding the result at precision level of a day. (For example, a truck driver that may drive 8 hours a day and has a driving duration of 10 days would result in a start/end date difference of 30 days. The start and end date of the activity may be described using the time unit 'day' although a time stream with a second precision was used for scheduling). Thus to some extent, the use of several different time units (also within a single activity) may make sense and will lead to consistent scheduling results. In general, the use of several different time units may lead to strange-looking results that can only be understood by accounting for the fact that the result dates are time slices and not simply time stamps.

Note that to get consistent results when using several different time units for objects of a single activity (*SUNIT, DUNIT, EUNIT*), the factory calendar, the time stream and the time zone assigned to the activity have to be consistent. This is the case, for example, if transaction /SAPAPO/CALENDAR is used to define the time stream using the time zone and factory calendar assigned to the activity.

### 3.8.2 Scheduling a Single Activity With External Processing

External processing may be used to:

a) Determine both start and end dates of the activity (2nd grade processing)

Or

b) Determine the start date of the activity (1st grade processing)

Or

c) Determine the end date of activity (1st grade processing)

During the pre-step of the SCM scheduling controller, each ABAP-OO class to be used for external processing (according to the evaluation of the dynamic scheme) is asked to determine the details for external processing. This allows the external class to specify at most one out of the three possibilities a), b) and c). This information is stored in the graph objects of SCM scheduling (Class /SCMB/CL_SC_GRAPH) and is thus transferred to the scheduling calculator. Therefore the calculator has the information as to whether 1st grade or 2nd grade external processing will be applied.

If a) is applied to an activity, the algorithm described in section 3.8.1 is not performed. The interface of external processing will include the complex date objects so that the time slices of start and end date of the activity can be specified during external processing. Thus, external processing may override the settings of the precision profile for start and end date of this activity. The external processing interface will also have a flag that may indicate that external processing determines a single time stamp only for each date. In this case, step 4) and 5) of the algorithm of section 3.8.1 will be performed for the time stamps determined in external processing in order to determine a time slice according to the time unit setup of the precision profile.

In the case of b) determination of the start date by external processing will be carried out as a preceding step 0) to the algorithm described in section 3.8.1. The external processing interface will have a flag that may indicate that external processing determines a single time stamp only and thus the time slice has to be determined by processing step 4) of the algorithm. Otherwise step 4) is not processed and external processing overrides the settings of the precision profile. In addition, the interface will have a flag to indicate that step 1) should not be done. Thus the start date as determined by external processing is not validated for active working time according to the calendar assigned to the activity. This is needed, for example, if the start of the transport activity should be aligned according to a route schedule. The start date as determined by external processing will not be changed within the core-scheduling framework.

In case of c) determination of the end date by external processing will be carried out as an additional step between step 2) and 3). The end date is thus derived by first scheduling according to the duration of the activity and than by applying additional constraints due to external processing. As with case b) the two flags exist to inhibit the processing of steps 3) and 5).

In the case of backward scheduling the algorithm described here and in section 3.8.1 is used in the same way.

### 3.8.3 Scheduling Follow-up Activities

The algorithm of single activities can be connected so that in the case of forward scheduling, the result of step 4) serves as the entry point date for the scheduling of a follow-up activity.

Note that the start time stamp of the time slices is always used. In the case of forward scheduling the start of the time slice of step 4) is used as entry time stamp for scheduling follow-up activities. In case of backward scheduling, the entry point time stamp is the start time stamp of the time slice that describes the start date of the activity that is the successor activity in time.

### 3.8.4 Basics of Graph Propagation

A graph consists of nodes (vertices) and lines (edges) that connect vertices. If for all pairs of nodes (*n, n'*) a connecting path exists (by propagation along lines) the graph is called a coherent graph or a network.

A line may have a direction, which means that propagation along the line is allowed only in one direction. Such a line is called an arrow. A digraph is a graph with lines that all have a direction (lines without direction do not exist).

Lines that connect nodes (*n, n'*) are adjacent to *n* and *n'*. A digraph may be defined by a list of adjacent lines for each node, the *adjacent list.*

Two basic methods *depth first search* (DFS) and *breadth first search* (BFS) exist to propagate through a graph.

In the case of *breadth first* search (BFS), all lines adjacent to the starting point node will be propagated first. This means that all nodes that are near to the entry point node will be visited first. In further iteration steps, all other nodes will be visited. The sequence depends on the number of arrows needed for propagation from the entry point to a specific node.

In the case of depth first search (DFS), an adjacent arrow of the starting point node is selected and the propagation occurs throughout the whole graph until an end node is reached. At each secondary node, a single adjacent arrow is used for propagation. An end node is a node that has no arrow that is still unprocessed. Such propagation along a path of several arrows is called a single propagation. After an end node has been found, the next adjacent arrow of the starting point node is used. When all adjacent arrows of the starting point node have been processed, the remaining adjacent arrows of the secondary nodes that are directly connected to the starting point are processed.

The paths of arrows that are used for the depth first search form the *DFS tree.* Arrows that end at a successor node of the DFS tree that has already been processed are *forward arrows.* Arrows that end at a predecessor node of the DFS tree (which has already been processed) are *backward arrows.* Backward arrows thus form a cycle in the graph.

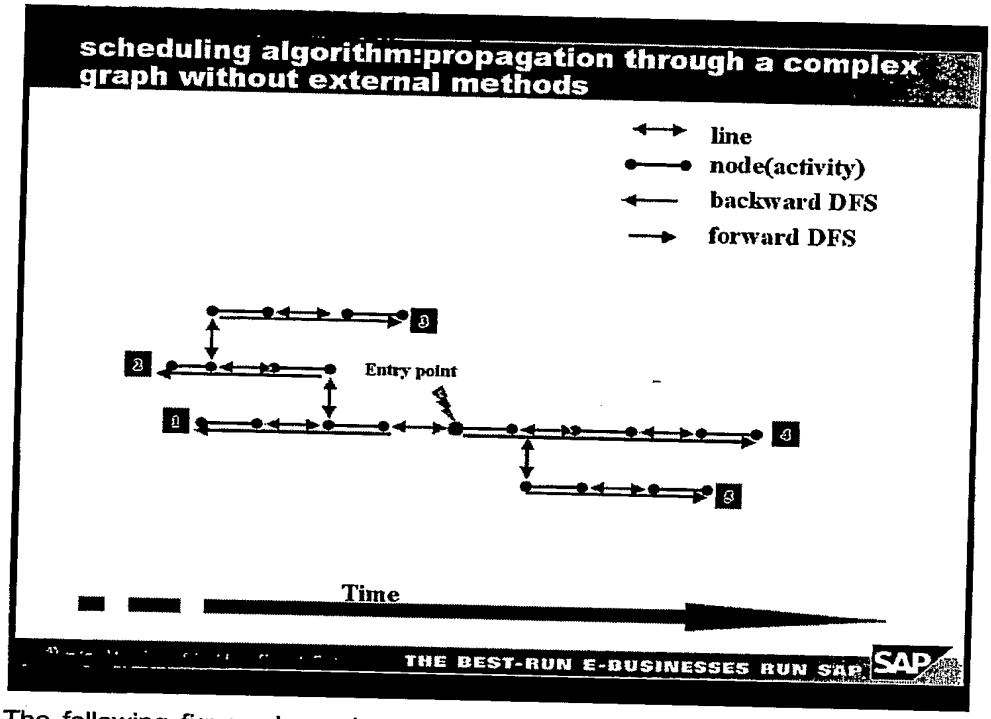
One may define a length of lines that connect the nodes. Thus propagation along an arrow (line) would cause a value increase on the length scale. It is also possible to define a negative length that results in a value decrease when propagating along the arrow.

### 3.8.5 SCM Scheduling in the Framework of Graph Propagation Basics

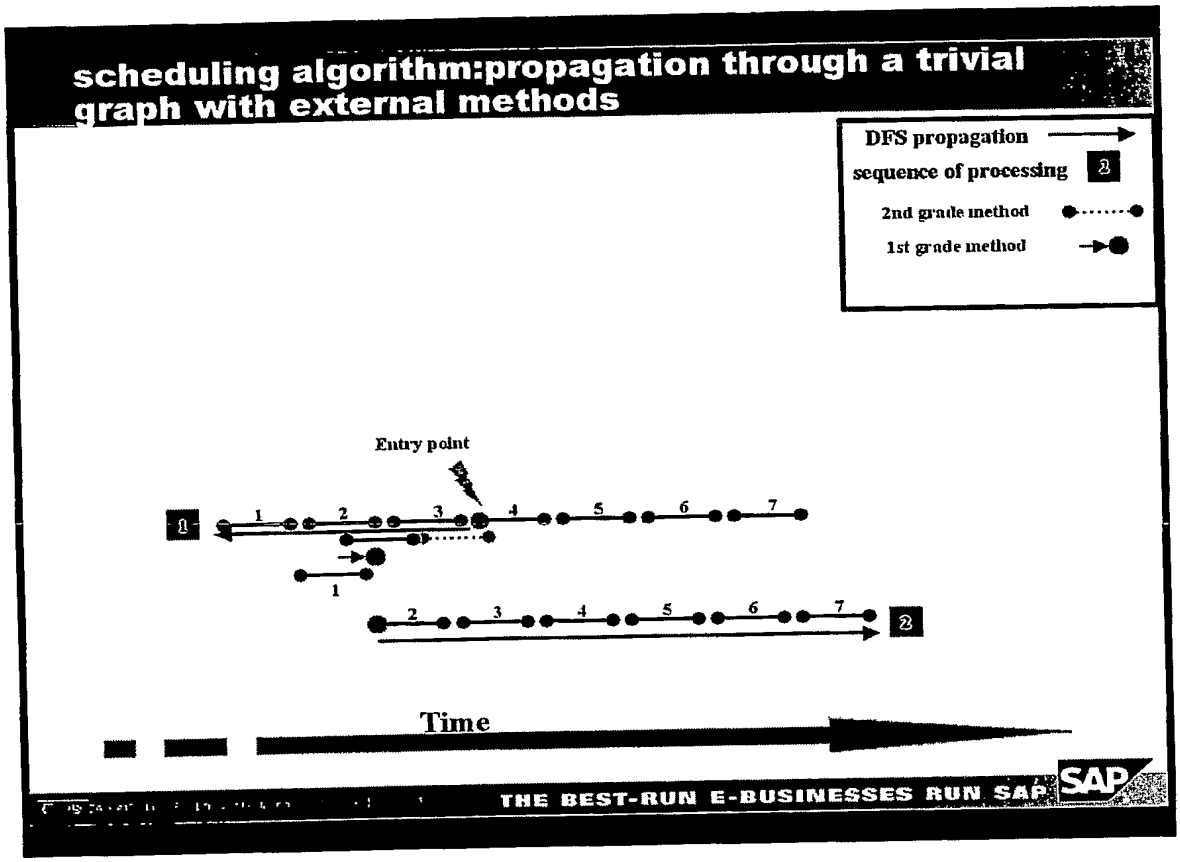
The network of activities (workflow steps) as defined by BPM form a graph. The activities are the nodes of the graph. The BPM data model comprises these nodes and the lines that connect the nodes.

SCM scheduling will use the DFS algorithm because it is important to at first perform backward scheduling to the very first date present in the network. This arises from the possibility to constrain the earliest allowed date and from the possibility that in backward scheduling, an activity can be shifted towards the future due to the results of external processing. For example, an ATP check on the start of picking might postpone the activity to a later date during backward scheduling also. If one of these actions takes place, the complete result of the backward scheduling performed so far becomes invalid and the scheduling has to be restarted with a new entry point. If a backward scheduling (starting from the entry date) is performed using the DFS algorithm, the number of restarts of the scheduling is minimized.

The BPM data denotes the time relation of the activities by defining a line by a pair of (predecessor node, successor node, (n, n')). SCM scheduling will perform a DFS by first carrying out a backward depth first search with a subsequent forward depth first search. In the backward depth first search, all nodes (activities) that can be accessed by starting with a backward (in time) arrow from the entry node will be visited. Backward depth first search scheduling may also comprise paths that include propagation forward in time, see DFS tree path number 3 in the following figure.




The following figure shows how a simple graph of two branches is scheduled with external processing. First a backward DFS is carried out. Within this backward DFS a 1st grade external processing delays the start of the second activity. Due to this delay, the already scheduled activities 3 and 2 are invalidated and the forward DFS is started with a new entry point (start of activity 2).

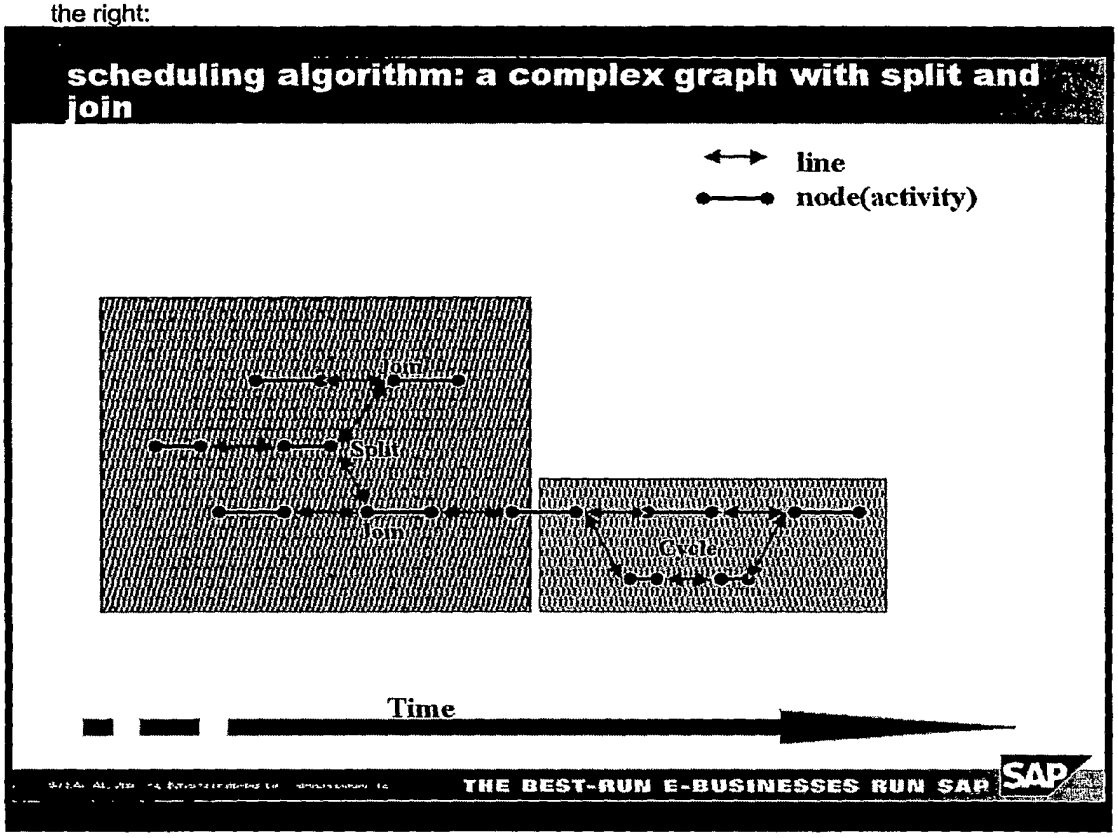



### 3.8.6 Restrictions of the Graph

SCM scheduling will not support all kinds of graphs. The following restrictions apply:

- Only coherent graphs are allowed. SCM scheduling will not schedule several independent networks in a single call.
- A length of lines is not supported. You may define the length of a line by assigning duration to it. Thus the end date of a predecessor and start date of successor activity could differ by an amount of time due to the network definition. Also an overlap of activities would be possible due to a negative length.
- The nodes of the graph are the scheduling activities, not the scheduling dates. Thus the graph always describes the relation of start-/end- date of activities of successor-/predecessor-activities. It is not possible to link an end date of an activity to an end date of another activity. A start-start link is not possible.
- Graphs with sequential activities that split up into several parallel branches that join into several follow-up activities are not supported by SCM scheduling in the first release. Graphs that contain parallel activities that do not form a cycle are allowed. In the terminology of graph propagation, all graphs containing forward- or backward arrows (see the definition in 3.8.4) are not allowed in the first release of SCM scheduling. In the case of these graphs, scheduling results of paths of the DFS tree may become invalid due to subsequent scheduling of forward-/backward arrow paths that result in incompatible dates of activities being directly linked together. For example, a follow-up activity, which was already scheduled due to

propagation along a DFS tree path, may have an earlier start date than the end date of the last activity of a forward arrow path that is the predecessor of the follow-up activity. This incompatibility would have to be resolved by additional iteration steps that recalculate the DFS tree path. This feature will be a matter of a future release of SCM scheduling. Here is an example of such a complex graph with allowed split/join and a non-allowed section on the right:




- Rules are not supported at the split/join points of a graph. The graph propagation algorithm only knows the predecessor-successor relations of the activities. For example, you may think of rules such as: Start the follow-up activity of a join if one of the predecessor activities is completed. This kind of rule is not supported. The graph propagation and the predecessor-successor relations imply that a follow-up activity of a join will have a start date later than the latest end date of predecessor activities.

### 3.8.7 Depth First Search Propagation Algorithm

The following is a sketch of the basic algorithm:

1) Definitions:

*B* is the array of nodes that were visited during depth first search

*R* is the array of nodes that have arrows that are still unprocessed

*N* is a node

*PN* is an array that comprises all unprocessed arrows of the adjacent list of *N*

Interface of procedure 'propagate':

Entry point node b, *PN* for all *N (list of adjacent arrows of N)*

Coding of procedure 'propagate':

```
Initialize B and R                (reset)
Add b to B
Add b to R
WHILE (R is not empty)
  Select node N out of R
  IF PN is empty
    Delete N from R
  ELSE
    N' = PN (first element).end     (propagate next arrow of list of unprocessed adjacent arrows)
    delete PN (first element)       (mark arrow as used!)
    IF N' is not element of B
      Add N' to B                   (mark N' as visited node)
      Add N' to R                   (include N' into the array of nodes that might have
                                    unprocessed arrows)
    ENDIF
  ENDIF
ENDWHILE
```

For a depth first search: 'Select node out of *R*' has to be implemented as last in first out (LIFO). For breadth first search: 'Select node out of *R*' has to be implemented as first in first out (FIFO). The LIFO implementation can be carried out using an internal table for *R*. Adding an element to *R* can be carried out by an insert table index 1, the "Select node out of *R*' can then be carried out by a read index 1.

In order to separate into backward and forward DFS scheduling, the procedure 'propagate' has to be processed twice. For backward DFS the list of unprocessed adjacent arrows of the entry point node (activity) will only comprise the backward links. For forward DFS the list will only comprise the forward links. A backward link of node *N* is a line with node *N* as successor node.

In addition to the general split into backward/forward DFS propagation, all lists of adjacent arrows *PN*(n) will be sorted in such a way that the backward links are first. The above algorithm that selects the next node by looking at the first item of the list of unprocessed arrows with code *N'* = PN(first element).end will first execute the DFS paths directing into the past. This is shown in the above figure of propagation via a complex graph (see 3.8.5) where path 2 is propagated before path 3.

The scheduling of individual activities will be done when the node (activity) is marked as visited in the above-described DFS algorithm.

### 3.8.8 Overall Scheduling Algorithm for a Single Scheduling Request

The DFS algorithm is only a part of the complete scheduling algorithm, because in addition to propagating via a graph the scheduling has to fulfill several features (mentioned at the beginning of section 3.8).

Here is the rough flow control of the complete algorithm (see below for flow control figures flow_control_fig).

- While (no valid results and convergence strategy not yet done)

-Loop on entry points (last one is the nominal entry point)

- Backward DFS propagation from entry point.

  If during backward propagation in time a delay occurred due to external processing (such as an ATP check), the delayed activity that was calculated at last will be the entry point of a subsequent forward DFS propagation. A complete path (branch) will be processed to its end before switching to the forward DFS.

  If during backward propagation in time a fixed date cannot be met by the scheduling, this will be analogous to delays due to external processing. If the fixed date is earlier than the scheduled date, no special handling will be carried out. In the case of backward processing in time, the earlier date will be taken. In case of forward processing in time, an early fixed date that cannot be met will cause a backward DFS propagation starting at the fixed date.

- Forward DFS propagation from entry point or an activity that was delayed due to external processing in the backward DFS. Forward DFS may include a scheduling backward in time. If during this backward scheduling a delay occurs (due to external processing) this will create the need for a forward DFS starting with this delayed activity.

- Push/pull optimization: Redo scheduling starting from the date that should be fixed in the optimization. Pull/push will start backwards/forward DFS in one direction only (starting from the date that is specified in the optimization request).

  Check whether this optimization date is unchanged in the new result. If it has changed, the result obtained before optimization is taken. Optimization failure can occur due to activity delays created in external processing.

- Check on earliest allowed date. If needed, redo complete scheduling with a forward DFS starting at the earliest node (start date of an activity) as calculated in the previously

preformed scheduling.

- Check whether the request is met by the processing of this entry point. (->valid result)

  If the nominal entry point has been processed (and no error due to convergence has occurred), this is always a valid result

- End loop on entry points

- Exception handler for endless propagation: Ensure consistent result data and/or invoke convergence improvement

-ENDWHILE (no valid result, and convergence strategy not yet done)

**Note 1:**

Endless propagation through the graph may occur. These endless propagations have to be detected. Detection of endless propagation will be made based on the total number of processed single activity schedulings. Every time an activity is scheduled, this number is increased and checked for:

Number of single activity schedulings > (N*N) * (number of entry points) * P * M * F

where N is the number of nodes (activities),

P = 1 in case of no push/pull optimization and P=2 in case of optimization

M = 1 in case of lower limit for allowed dates is given and M = 2 if a limit is given

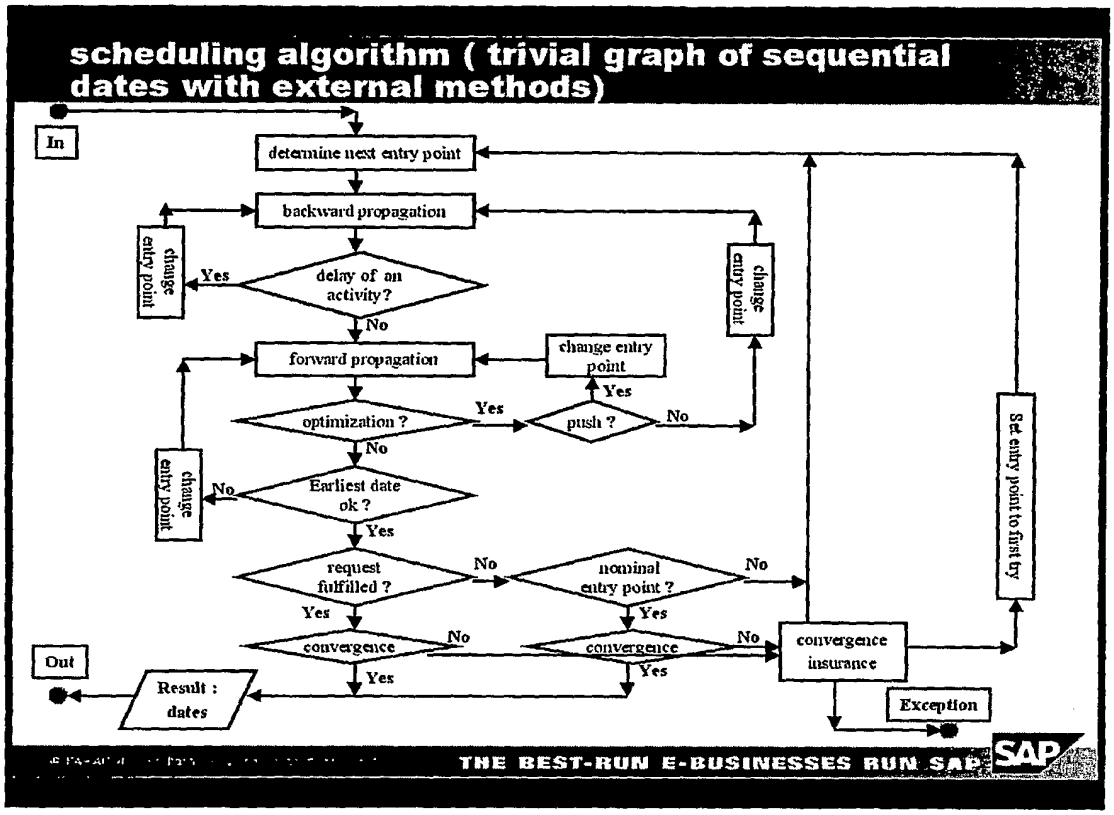
F is the number of externally given fixed dates

If the number of single activity scheduling gets too high, the complete scheduling is stopped and an exception is raised from within the algorithm. This exception is caught by the exception handler in the outer layer of the algorithm that ensures consistent result data and may invoke a convergence improvement. Convergence improvement will not be implemented in the first release of SCM scheduling but is foreseen in the architecture of the scheduling algorithm. It is foreseen that convergence improvement may change the scheduling problem slightly so that a solution for the problem may be found.

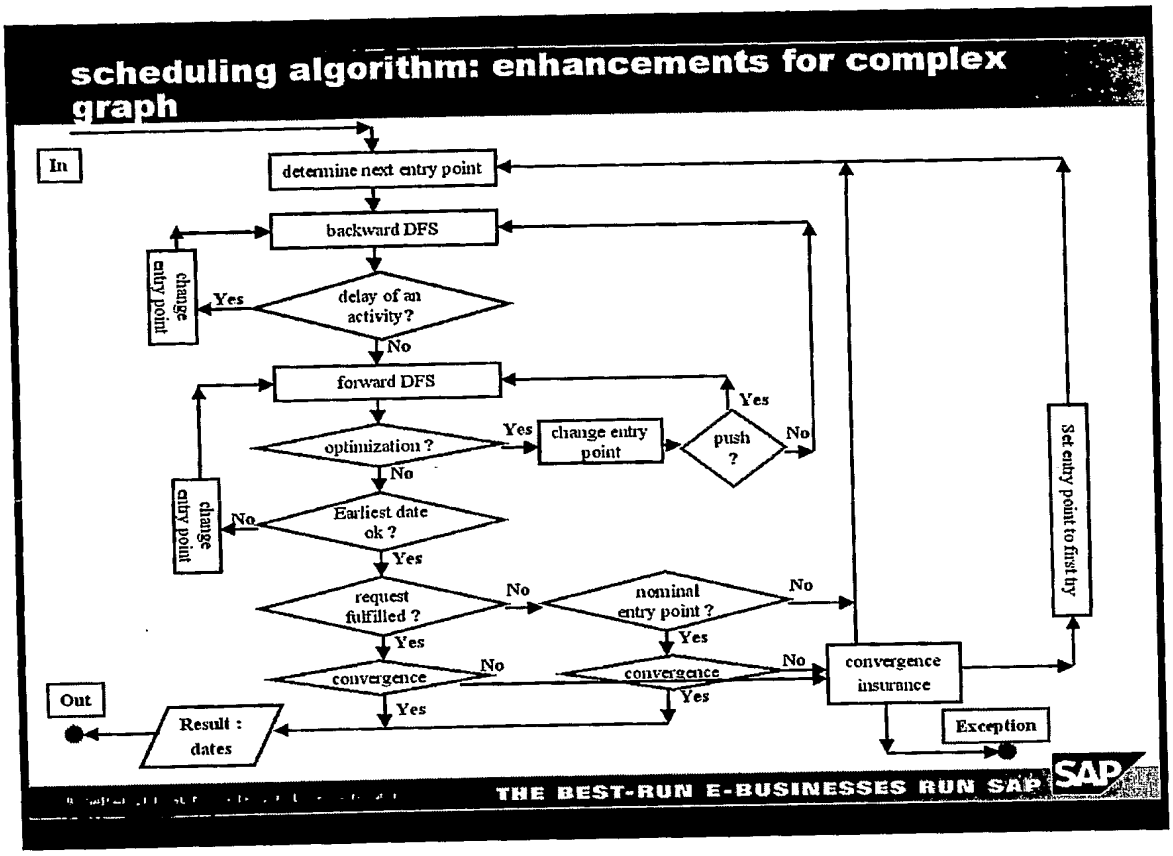
**Note 2:**

Detection of endless propagation may be due to the fact that no valid result for the scheduling problem exists. There are several ways of setting up a scheduling problem with the various features described at the beginning of this paragraph (earliest date, push/pull, and so on) that has no valid result. Some of these unsolvable settings may be easily detected, such as setting a fixed date that is earlier than the earliest allowed date. These easily detectable errors will be checked for right at the beginning in */SCMB/CL_SC_ENTR* - this validates the interface data before the controller pre-step (that is, without detailed knowledge of the actual scheme and network). In addition, at the beginning of the calculator */SCMB/CL_SC_CALC,* before performing the real scheduling algorithm, a rough check on the feasibility of the problem can be applied. This check may calculate the time difference between certain fixed dates and compare it to the sum of the duration of the activities between the fixed dates. The difference of the fixed dates has to be greater or equal to the sum of the activity duration. Because this second (smarter) kind of check will be more time-consuming, the validation procedure at the start of the scheduling algorithm will be optional.

The following figures show the flow control of the algorithm. Backward propagation denotes backward scheduling in a simple linear graph.




A more complex graph can be scheduled using the same algorithm with the DFS algorithm. Note that the flow control figure does not contain the fixed dates feature. This will be handled in the same way as the delays due to external processing (see textual description of the algorithm above).




### 3.8.9 Use of Fixed Dates

When the algorithm propagates through the graph of activities, real scheduling of an activity is only performed if one date (start or end) of the activity is not fixed. Start and end date of the activity are always rounded (see steps 4 and 5 in 3.8.1). Fixed dates are thus not treated as absolutely fixed but are slightly modified by taking the precision of the dates into account.

During processing through the graph it may be that the given fixed dates are not consistent with the part of the graph of activities scheduled so far. For example, a scheduling may start with entry point date 'start of picking' and result in an 'end of loading' date that is later than the given fixed date 'start of transportation'. This inconsistency is treated in the same way as inconsistencies that might occur due to external processing (see 3.8.8). A new scheduling is started by taking the fixed date as the entry point date.

Time gaps between fixed dates and dates of adjacent activities are not resolved (optimized) by the scheduling algorithm.

1st grade external processing is not performed for fixed dates (even if it is set up in the scheduling scheme). 2nd grade external processing is not performed for activities that have one or more fixed dates (even if it is setup in the scheduling scheme).

### 3.8.10 Splitting Due to External Processing

Splitting the request for scheduling into several results with different dates will be allowed for external processing. The first result line will be used for further processing of the graph propagation performed so far. The other results of the external scheduling will be used as entry points of several independent processings of the complete scheduling algorithm. The scheduling does not perform any type of correlation of individual scheduling results that occur in the case of splitting a single scheduling request due to external processing.

Further processing of the partly executed graph propagation is perfect if the external processing does not change a property that is relevant for the determination of scheduling master data. For example, such a changed property could be, in the case of ATP, the quantity, material or location. Change of properties will be discussed in the next section.

### 3.8.11 Change of Properties Due to External Processing

The interface for external processing will include an import table that holds the table of internal properties that were used in the pre-step of the controller for the determinations within the dynamic part of the scheme. The interface for external processing will include an export table of changed properties (this may be any property from the import table of properties).

Property changes may be for example, a quantity reduction in the case of an ATP check, a material/location substitution in the case of a global ATP check, substitution of means of transport in the case of a check against transport resources. Note that the concept of changed properties is not limited to any specific properties. Any property may be changed by external processing.

If the table of changed properties is not empty, a check will at first be performed to evaluate if the dynamic part of the scheme depends on the changed property or not. This will be done by a controller method that checks whether the dynamic part of the scheme does not directly depend on one of the changed properties. This controller method also asks all of the classes, which are set up in the dynamic part of the scheme, to confirm that their determinations do not depend on the changed property.

If one of these checks is negative, a complete re-evaluation of the dynamic scheme has to be made in order to re-determine the scheduling master data (calendar, duration, and so on.). This will be carried out by a controller method. After this re-determination, the scheduling algorithm will start at the very beginning using the result of the previous external processing (that changed the properties) as the entry point date to start the algorithm. If, in addition to a change of properties a splitting also occurred, several independent processings of the complete scheduling algorithm will be performed with the splitting data (obtained from the previous external processing) as the entry points.

### 3.8.12 Executing Multi-Scheduling Requests

Up until now only the possibility of scheduling a single request that is passed to the scheduling interface method /SCMB/IF_SC_SCHED=>schedule has been discussed. This method will, however, be able to schedule several requests in a single service call.

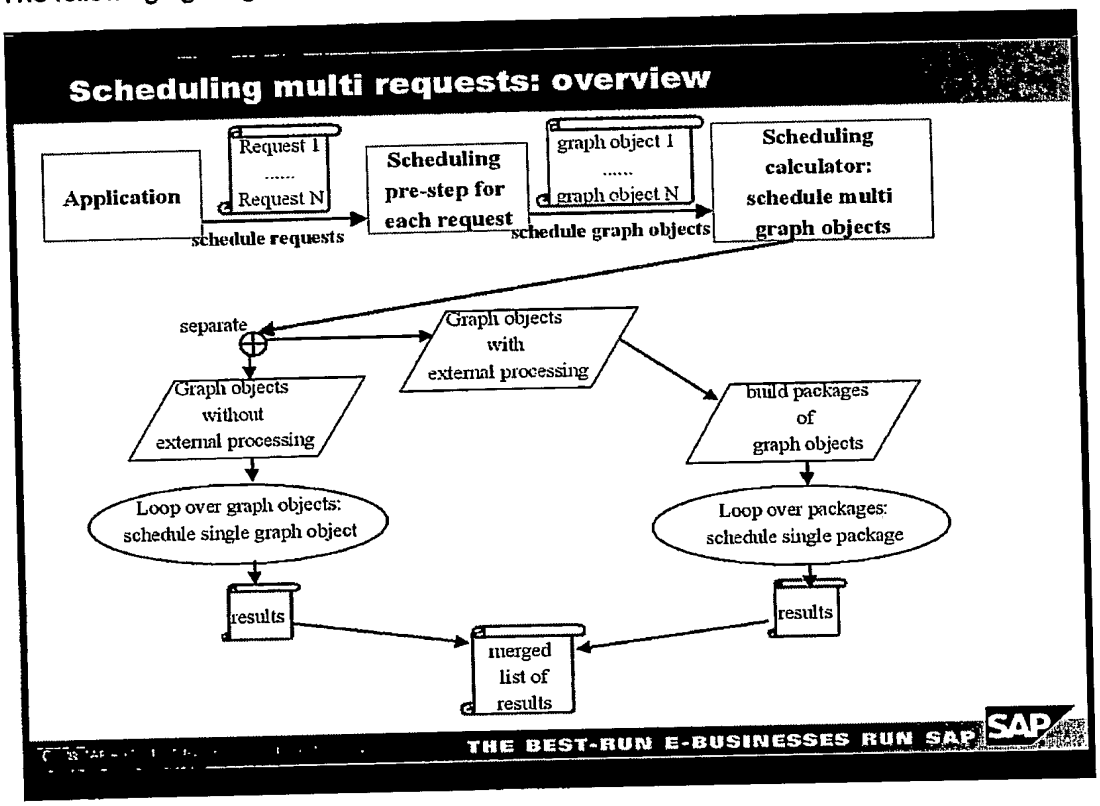
In the case of multi requests, the controller /SCMB/CL_SC_CONT will carry out the pre step for each request independently in a loop. This will end up with a graph object instance of class /SCMB/CL_SC_GRAPH (which holds all information needed to carry out basic scheduling

through a graph of activities) for each individual request. These objects are passed to the basic calculator /SCMB/CL_SC_CALC->calculate.

The calculator will check for external processing in each graph object. Those graph objects that have not set up external processing will be processed first in a loop analogous to the scheduling of an individual request.

All graph objects that have identical ABAP-OO classes set up for external processing (irrespective of the activity for which this class is used) will be put together in a package of graph objects. Note that created several packages, which are disjunctive with regard to ABAP-OO classes used for external processing, are created. Each package is processed individually. By means of a package, all external processing done by a certain ABAP-OO class can be accumulated. The need to accumulate several requests for external processing arises from performance aspects as well as from the fact that external processing may perform a correlation of several requests (like GATP delivery correlation as of APO 3.1). When all packages have been processed, the calculator /SCMB/CL_SC_CALC is finished.

The following figure gives an overview of the scheduling of multi requests:




### 3.8.12.1 Scheduling a Package of Requests With External Processing

The calculator /SCMB/CL_SC_CALC starts the scheduling of the first request (graph object) of the package using its own private method DFS_PROPAGATION(). This method makes use of the activity objects that perform the scheduling of a single activity once a node (activity) is reached in the DFS algorithm. The activity objects will use instances of ABAP-OO classes for external processing. During the scheduling of a package of requests, the methods for external processing will have an indicator flag set that gives the external object the information that several calls may be accumulated (by the ABAP-OO class used for external processing). The external object may then decide on its own if it has the ability to accumulate several calls. If it does

not accumulate but performs the external processing immediately, the scheduling algorithm continues as described above in section 3.8.2. If the external ABAP-OO class has the ability to accumulate several calls, it has to buffer the request. In addition, the external object has to set up an event handler for event DO_EXT_PROC of class /SCMB/CL_SC_CALC. The method used by the activity objects to carry out external processing will then return a flag that indicates that the external processing request has been buffered and was not processed immediately.

At this point, the scheduling algorithm of this request has to be stopped and the complete state of the DFS propagation and the scheduling of the single activity (where the external processing has to be applied) have to be captured. This will be done by creating objects MPO_DFS_PROPAGATION_STATE that are private objects of calculator class /SCMB/CL_SC_CALC. The captured state will guarantee the possibility to re-start the scheduling algorithm for this request at the same point where it has been stopped.

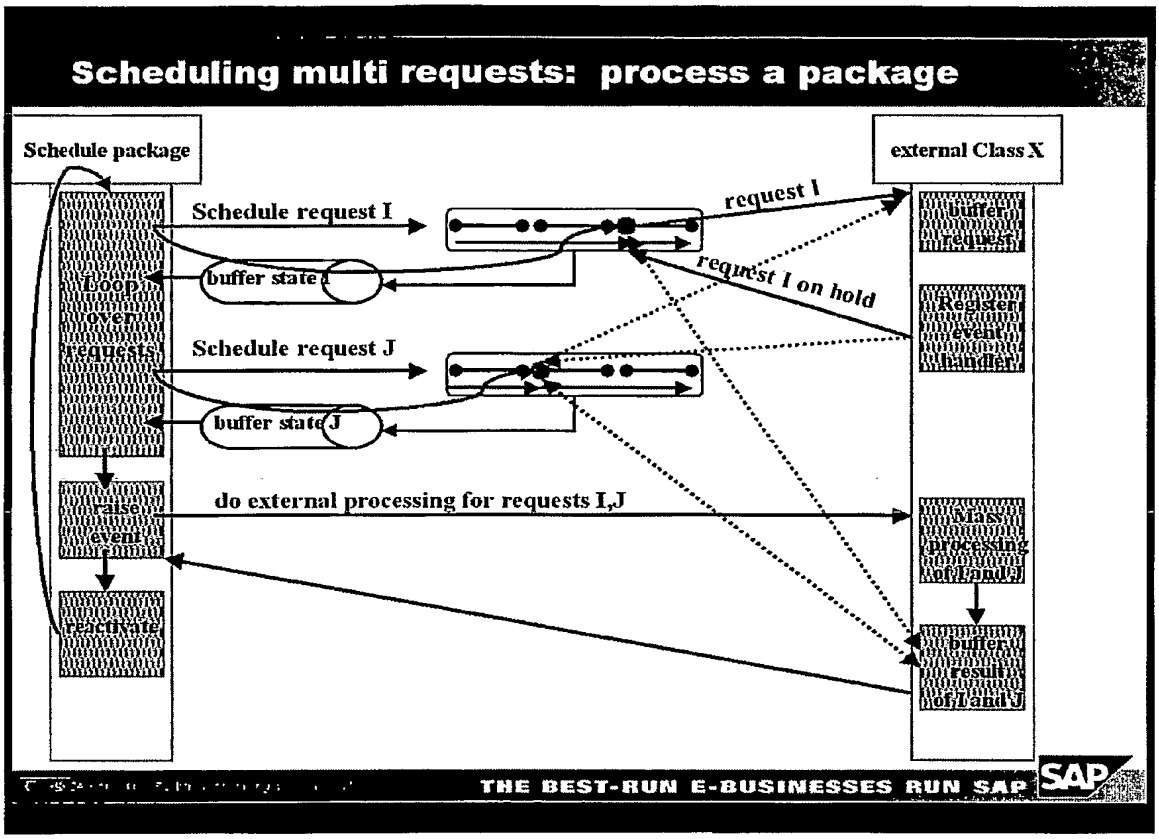
After the state has been captured, the program flow will exit the scheduling of the individual request and will start scheduling another request of the same package. When all requests of a package have been processed, there are two possibilities for each request. Either it is already scheduled completely because no accumulation of external processing was carried out for it or the scheduling of this request is on hold. Requests on hold can be identified by the fact that they have an associated state object that was created when the external processing request was buffered. If at least one request on hold exists, the calculator will raise the event DO_EXT_PROC that will start the handlers of the external ABAP-OO classes. These handlers will execute the mass execution of the external processing for all buffered requests for external processing of a package. The handlers will buffer the result of the processing in the external ABAP-OO class. In ABAP-OO runtime the handlers are executed synchronously. In this way, the SCM scheduling calculator steps back in when all handlers (all external mass processing) is complete.

Calculator /SCMB/CL_SC_CALC now has to re-invoke the scheduling of the scheduling request(s) that are on hold. This can be done by re-starting the DFS algorithm with the captured state object. Now the result(s) of external processing can be retrieved from the external class and the scheduling of these request(s) may be completed.

Note that the mechanism used to capture the state of the scheduling algorithm and put the scheduling for a request to status 'on-hold' may occur several times during the scheduling of a network of activities.

Note also that inside SCM scheduling there is nearly no coupling of the several scheduling requests that carry out external processing by means of single mass processing. The only existing coupling inside SCM scheduling is the package structure.

The following figure shows the concept for accumulating individual requests of external processing:




## 3.9 Explanation Service

### 3.9.1 Explanation Service Architecture

A sketch of the basic architecture has already been discussed in section 2.2.3.2.

The explanation service will comprise a view controller class /SCMB/CL_SC_VIEWC and the following standard views:

1) An ALV list that gives an overview of the activities and their date types, date values and time zones. External processing of dates or of the complete activity is also indicated. The entry point date is indicated as well as whether the entry point date has been changed.

2) An ALV list that shows the details of activities. Duration, factory calendar/time stream and locations/partners of the activity are displayed. This list of details is only available for those activities that were not completely scheduled by 2nd grade external processing. The details of an activity also include information on the determination procedure of master data (whether it was done via direct value mapping of properties provided via the interface or whether it was done by means of an ABAP-OO class).

3) Views that can be provided by the ABAP-OO classes that were used for master data determination. They might provide a protocol of these determinations. An example would be a protocol of the condition technique or a protocol of the BRR.

4) A view that is provided by the classes of external processing. This view can be invoked for each external processing of a single date or a complete activity. Thus the details of external processing might be shown in the framework of the overall SCM scheduling.

5) A SAP Gantt chart graphic (made by SAP bar chart graphic) that shows the activities scheduled on a time scale. Change of dates will not be possible. The graphic will be a display tool only. The activities are shown together with the calendar and constraints such as the fixation of dates.

6) An ALV list that shows a log of the scheduling algorithm, including the graph propagation. It is thus possible to see how the algorithm propagated through the graph and how the algorithm performed several iterations.

The number of views is not static but depends on the dynamic part of the scheduling scheme (because external processing may also enable an explanation service view). A dynamic view controller will be created to allow you to switch from view to view. A split screen will be used for the view controller. A tree control on the left side will allow for fast navigation through the available views. The views will be shown on the (larger) right part of the split screen. Navigation between the views can be done using the tree and/or by buttons/double clicking on the views. The tree holds an overview of available views for fast direct navigation.

### 3.9.2 UI Formatting of Dates

In the scheduling explanation service the resulting dates will be shown to the users. Basis package SZTI contains ABAP classes to format the output of dates. Interface class `IF_TIMEFORMAT` with implementing class `CL_TIMEFORMAT_SIMPLE` will be used to format the date fields of the ALV lists shown in the explanation service. The chosen format will depend on the precision of the dates. Date, time and time zone will be shown for time units smaller than a day. Day of week, date and time zone will be shown for time unit 'day'. Day of week and date will be shown for greater time units. The start time stamp of the time slice (which describes the dates in case of a unit unequal to 'exact') will be taken for the UI presentation.

These formatted dates (DDIC: TIME_IO) will be contained in the scheduling result too. In this way, applications that use SCM scheduling can directly create a formatted output of the result.

### 3.9.3 API for UI of Explanation Service

The UI presented in section 3.9.1 will be technically based on ABAP Dynpro technique because it makes use of Basis/ABA tools (ALV Grid Control, Basis front end services, and so on...) that run in the Dynpro only.

Some applications need to present the explanation data not in a Dynpro-based UI but based on other technical implementations such as Business Server Pages (BSP). In order to cover these requirements the explanation service will have an API that enables you to retrieve all SCM scheduling explanation data. This SCM scheduling explanation data also includes the information on the ABAP-OO classes that were used in the processing of the scheduling request (according to the settings of the scheme customizing data). The class method /SCMB/CL_SC_SCHED=>EXPLAIN can be used for this data retrieval. This method will have an input control parameter that indicates whether an UI should be created internally (as described in section 3.9.1) or if the method will be used to provide SCM scheduling explanation data only. The method will have an export interface which will provide all SCM scheduling explanation data with a calling application. This application may then invoke any kind of UI for the presentation of the data. For example, this application may have its own set of BSPs that are used to present the SCM scheduling framework explanation data. The explanation data that is provided by the API will be structured analogous to the structure of the several views that were

presented in section 3.9.1. Thus the several views that can be shown based on Dynpro technique can easily be re-implemented using another technique (BSP, for example).

The SCM scheduling explanation data will include the list of ABAP-OO classes that were used during the processing of the scheduling request. It will be indicated for each ABAP-OO class if this application component supports an explanation UI by itself to present its internal data (for example, this could be a log of the master data retrieval from condition technique). It will also indicate which technique the internal UI is based on (Dynpro, BSP, and so on). If the overall application uses BSP it might use the internal UI (also based on BSP) of those application components. The export interface of method /SCMB/CL_SC_SCHED=>EXPLAIN will contain a reference to a BSP for each ABAP-OO class used in the scheduling processing. This reference to a BSP has to be provided by the ABAP-OO class that was used in the scheduling processing. This reference can be used to dynamically switch to the next BSP page that will then be a page that presents the data of the ABAP-OO class. The technical type of this reference will be a URL.

Note also that the BSP used to present the data of an ABAP-OO class used in the SCM scheduling framework has to be created/provided by the development team that creates the ABAP-OO class. This arises from the fact that the BSP has to be adapted to the data that is managed by this ABAP-OO class. Due to the fact that this ABAP-OO class does not belong to the generic core framework, SCM scheduling cannot provide this BSP.

Please note also that it is up to the calling SCM application to create a view controller to navigate and invoke the several BSP pages. The SCM scheduling explanation service interface only contains explanation data and URLs of BSPs that might be used to present the data.

### 3.9.4 UI of SCM Scheduling Explanation Service Based on BSP Technique

Views 1), 2) and 6) presented in section 3.9.1 can be easily implemented as BSP because they are simple lists of data (in Dynpro technique they are made by ALV). These BSPs can be developed within the development project of SCM scheduling (for example, Fulfillment Coordination). As mentioned in the previous section, the data of the explanation API will be structured according to the several views. Each data package (of a view) may refer to a BSP. By means of this reference the BSP that will be used to present the data can dynamically be found and used by the calling SCM application. An SCM application that uses the explanation service to retrieve the data may use these references in order to invoke a UI (based on BSP). This interface presents the data of a certain data package. This is analogous to the technique that can be used to present the UI (BSP-based) of ABAP-OO classes used during the scheduling.

## 3.10 Scheduling Post Service

The scheduling post service will manage the saving of scheduling explanation data to the database. In addition it will allow you to save data of external ABAP-OO classes that have been used within the scheduling framework.

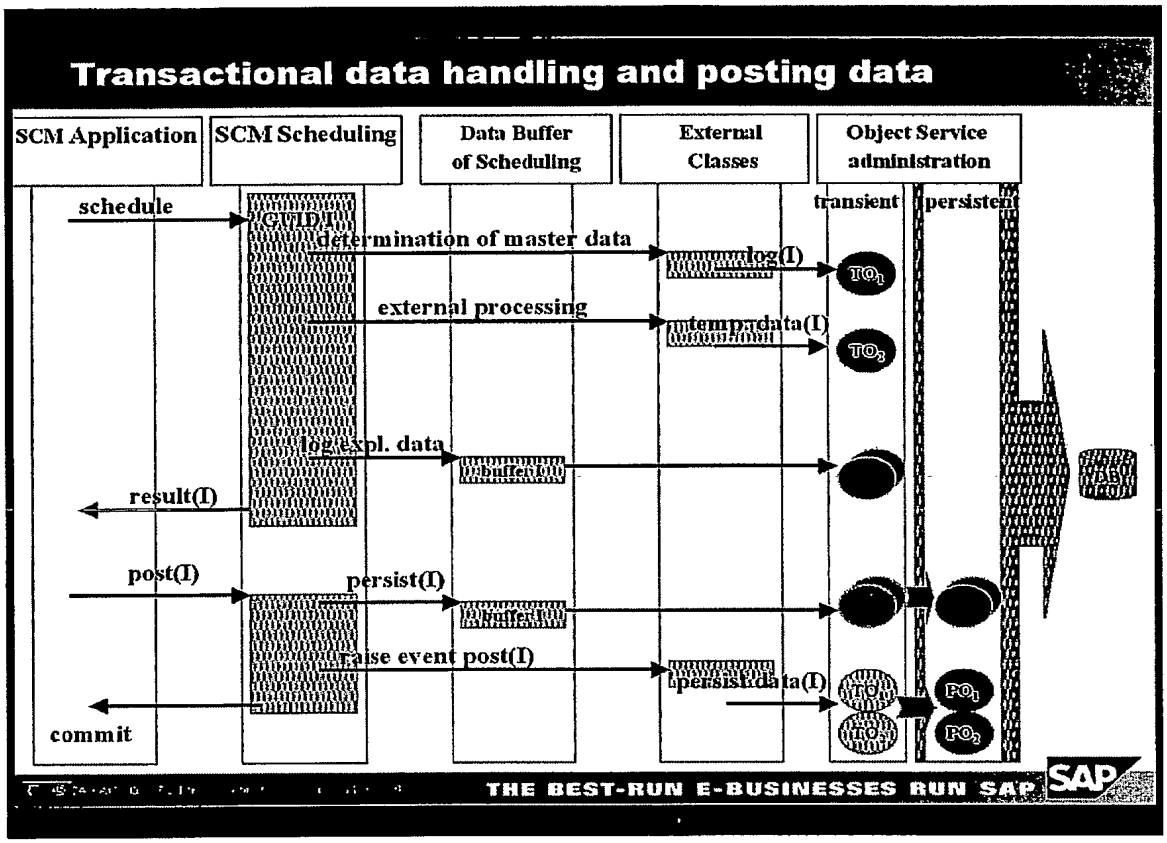
SCM scheduling will provide a post service that might be called by applications during their saving procedures. Note that the scheduling post service has to be called explicitly before the commit (work) statement! The post service itself will not have a commit (work) statement.

The scheduling post service is a class-method of /SCMB/IF_SC_SCHED. Each scheduling service result can be identified by a global unique identifier (GUID) that is passed in the result data of the scheduling service to the calling application. The scheduling post service interface will contain a table of GUIDS that denote the scheduling results for which the data will be saved. The SCM scheduling post service fetches the data from the transactional data buffer /SCMB/CL_SC_DATABUF (where the data was stored during the scheduling service call). The transactional data objects (their references are stored in /SCMB/CL_SC_DATABUF) will be implemented as transient objects according to the object service interface IF_OS_FACTORY of

SAP Basis layer of the class agent of a persistent class (see ABAP documentation on persistent classes for details!). During the SCM scheduling post service these objects will be converted into persistent objects. The persistent objects will automatically be written to the database (by means of the Basis object service) when a commit (work) statement is executed. If the application does not call the SCM scheduling post service, the objects that hold the explanation data will remain to be transient objects (according to interface IF_OS_FACTORY) and will not be saved to the database when a commit (work) statement is executed. Conversion of transient objects into persistent objects may be carried out using interface IF_OS_STATE of the persistent class. With method 'IF_OS_STATE->get' a copy of the transient object can be created. Afterwards the transient object can be released from the object service by 'IF_OS_FACTORY->RELEASE'. Then the persistent object can be created by 'IF_OS_FACTORY->CREATE_PERSISTENT' and filled by copying state object with method IF_OS_STATE->SET. The advantage of first creating transient objects instead of directly creating persistent objects is that per default no data is saved! The application has to explicitly trigger the data saving of SCM scheduling by calling its post service.

The scheduling post service will raise public event 'POST' of interface /SCMB/IF_SC_SCHED. This enables all instances of classes that were previously used for external processing (or determination of master data) to carry out their specific data saving. The instances of those classes will not be destroyed when the scheduling method is carried out if they subscribe to this event using ABAP statement 'SET HANDLER'. Each ABAP-OO class that is used in the framework of SCM scheduling may set up an event handler method for this event. The event handler method will have the GUIDs as parameters that identify individual scheduling service calls. These event handler methods may initiate a data saving by 'call function on commit', 'perform on commit' or by instances of persistent ABAP-OO classes. Due to the fact that in ABAP-OO a synchronous event handling is performed, a commit (work) statement, which may be placed after the call to the SCM scheduling post service, will always be executed after all event handlers have finished their processing. Thus the data saving of the application that makes use of SCM scheduling **and** the data saving of SCM scheduling **and** the data saving of objects of external ABAP-OO classes used within the SCM scheduling framework are in a single LUW. The advantage of coupling instances of external classes (which do not belong to the core framework of SCM scheduling) by means of a POST event is that SCM scheduling does not have to administrate the several tasks that have to be executed for data saving. Administration is carried out by ABAP run time and by the external classes that set up an event handler to carry out the data saving.

The following figure outlines management of transactional data and posting of data:


Transactional data handling and posting data
SCM Application
SCM Scheduling
Data Buffer of Scheduling
External Classes
Object Service administration
transient
persistent
schedule
GTID I
determination of master data
log(I)
external processing
temp.data(I)
log.expl. data
buffer(I)
result(I)
post(I)
persist(I)
buffer(I)
raise event post(I)
persist.data(I)
commit
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Annex 3

# Glossary

***SCHEDULING ACTIVITY*** *A business processes may be devided up into several process steps. Any process step that needs to be scheduled is a so called scheduling activity.*
***SCHEDULING SCHEME*** *From point of view of the scheduling a complete description of a business process is a scheduling scheme, which contains the information of all process steps relevant for scheduling (scheduling activities )*
***DATE TYPE*** *describes the business content of a date. For example start date of loading, end date of loading, etc.*
***DURATION TYPE*** *describes the business content of an activity for which the duration is used.*

# 1 Requirements (Overview)

## 1.1 Scope of Functionality / Description of the Business Processes (Business Context)

Given the fact that SC Basis scheduling will have to cope with a variety of different use cases by several business applications/processes, probably evolving in time and thus not fixed at specification date, the scheduling service has to be a generic service with the possibility of adapting it to the specific needs of an application/business process. So far, SAP standard applications of SCM products R/3 and APO use there specific scheduling functions, embeded and adapted to their special needs. While this provides best integration of the scheduling function with the specific application it is a drawback in cross-applications processes. Given that cross-application processes will be the majority of the business processes handled by SCM the generic layout of the SC basis services is a need. A goal of SC basis scheduling is a cross-application integration for best-of-bread process management.
One example business process where SC basis scheduling can be used is collaborative planning (SCC). Several different SC planning systems ( like SAP APO ) communicate via SCC there requirements ( given by demand planning/supply network planning ) to suppliers. SC Basis service scheduling will determine the logistic dates that have to be met by the supplier's logistics chain in order to fulfil the demand. The scheduling itself may come to the conclusion that the demand can not be fulfilled within the required timeframe and may publish a confirmed date which is a later point in time.
The business partner may publish his demand in terms of material availability date at his location ( that is the date when the whole goods receipt process is finished and thus the material can be used at the business partners locations for other business processes). Thus the scheduling has to take into account the goods receipt process also.
Given the above two sections the scheduling has to calculate the logistic dates of the supplier, the ( 3rd party ) transportation and the goods receipt processes of the ordering business partners.
In general, the logistic dates handled by SC basis scheduling will include all dates known so far in SAP R/3 and APO, but the service will extend the range of dates to free configurable dates. Possible business impacts of free configurable dates are:

- additional dates in the suppliers part of the logistics chain, like a description of a more detailed picking process by the steps picking from storage, unpacking, QM control, packing of transportation handling units ( instead of the classical R/3 and APO description by a single process step ‚picking' ).
- additional dates of the ( 3rd party ) transportation which might serve as check points for tracking or which might be used for describing a complex transportation by means of several partial-transportation which add up together to the whole chain from supplier to goods recipient.
- Additional dates of the goods receipt process may describe the goods receipt itself and additional work to be done before the material is released for free usage ( quality control, re-labeling, re-packing, etc. )

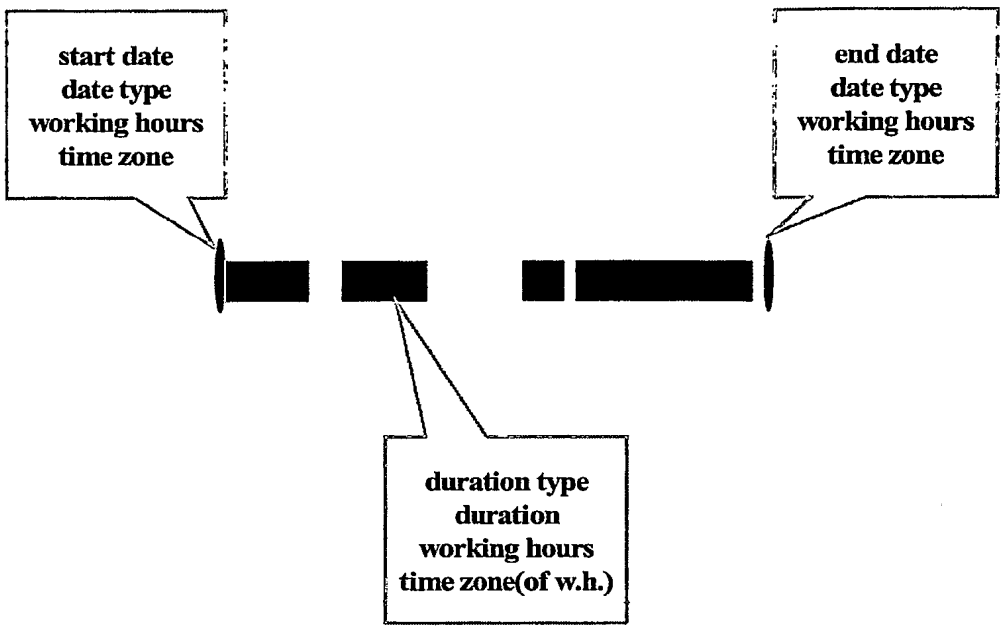
Existing SAP products R/3 and APO are not able to perform a scheduling with free configurable dates and are thus not able to schedule in detail the above given examples. Given the possible integration of SC Basis scheduling with SCM applications like SCC,SCFC and SC event management, a definition, tracking and control of supply chain processes in time will be feasible at a high level of accuracy.
In addition, the possibility of free configurable dates may lead to use cases of SC basis scheduling fully independent of SAP business processes and products existing so far. Free configurable dates will guarantee an open integration architecture to any business application which has a need for scheduling.
For example, one may think of a 3rd party transportation logistic partner who makes use of SCM and SC basis scheduling in order to schedule his transports in detail with the usage of the constraint dates ( i.e. loading date and goods receipt date ) given by the other two business partners. In this example, an availability check on means of transportation and means of loading/unloading seems to be necessary. This directly leads to the business requirement of a combined scheduling and checking.
SC basis scheduling will provide the possibility of defining check methods which are processed fully integrated within the scheduling. The little example above also indicates the need of a constraint based scheduling in a multi-partner environment. Thus SC basis scheduling will allow for a constraint based scheduling.
An example of the area of SC fulfilment co-ordination (SCFC) is the distributed order management (DOM) which is the logistic part of a sales order created by SAP CRM. The SCFC may perform a ATP check for the sales order, probably by calling into various subsystems ( like various SAP APO systems ). The SC basis scheduling will have to be invoked in order to calculate logistic dates which are not known by the various subsystems. This implies that SC basis scheduling has to be able to accept dates calculated by other services as fixed dates but enhancing this fixed part of the logistic chain by additional dates.
In contrast, SC basis scheduling may overtake the complete scheduling, which means determining all dates, also those which might be owned/known by the other subsystems(components). To enable SC basis scheduling for calculations of these dates, an interface for master data exchange ( duration ,calendar, time zone ) between the other components and the SC basis scheduling is needed. Thus, SC basis scheduling will make use of the same master data as local scheduling within the other components. This will lead to consistent dates, also from point of view of the other components.

# 2 Requirements (Details)

## 2.1 Description of the Individual Use Cases / Functions

### 2.1.1 Configurable definition of a scheduling activity

Each scheduling activity has to be defined before it can be incorporated into scheduling schemes.
Definition of a scheduling activity includes:



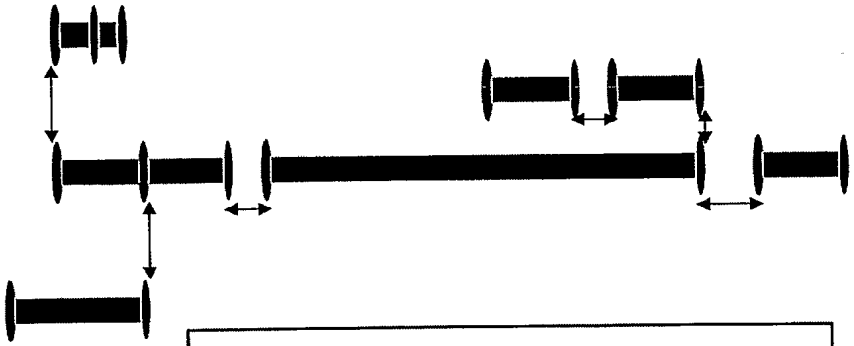

- business description: what is done by this activity
- short text description for GUI usage
- type of start date
- type of end date
- type of duration

SC basis scheduling will provide maintenance tools for the definition of scheduling activities. This will include function based tools which enable for maintenance triggered by other applications ( for example a business design tool of SCFC ). This definition will include a definition of date types and duration types that can be used by SC basis scheduling.
A scheduling activity is defined by the types of start date, end date and the duration type. All other attributes presented in the figure may be determined dynamically. Please note, that the effective time difference between start date and end date may be longer than the duration of the activity due to working hours. Start date and end date of the activity are dates of active working hours. The picture shows a schedule activity, the green bar with gaps indicates the active working hours.
Several different activities may have the same date/duration type. For example if a business process consists of loading at the shipping plant and a complex transport with several load transfer points the loading at the ship-from and the load transfer point may be represented by two activities which refer to the same date/duration type.

### 2.1.2 Configurable definition of scheduling scheme

The SC basis scheduling will be able to support free configurable scheduling schemes. A scheduling scheme defines a list of activities that have to be scheduled and defines their relation in time.
Example: a scheduling scheme may consist of the activities pure picking(fetch the material from storage), unpacking, quality check and packing of transportation handling units. Their relation in time may be defined thus that all activities are done in sequence, except the quality check, which is in parallel to the packing of transportation handling units. The definition of the time relation of the activities will be done by a link of the start of an activity with the end of a preceding activity or a link of the end of an activity with the start of a follow-up activity. The actual start-date and end-dates determined by the SC

basis scheduling of activities in sequence may be different due to different working hours assigned to the activities. Note that the restriction applies that parallel activities have only one point in time of synchronisation, start or end of the activities. Please note, that this definition of the relation of activities does not allow for a relation like: 'start of activity 4 not earlier than end of activity 3 and end of activity 7'. It is only possible to implement 'start of activity 4 not earlier than end of activity 3' like it is shown in the figure.
The activities which can be used for schedule schemes have to be defined as a scheduling activity. Definitions of schedule schemes may be done by inheritance. The package of SC basis scheduling will

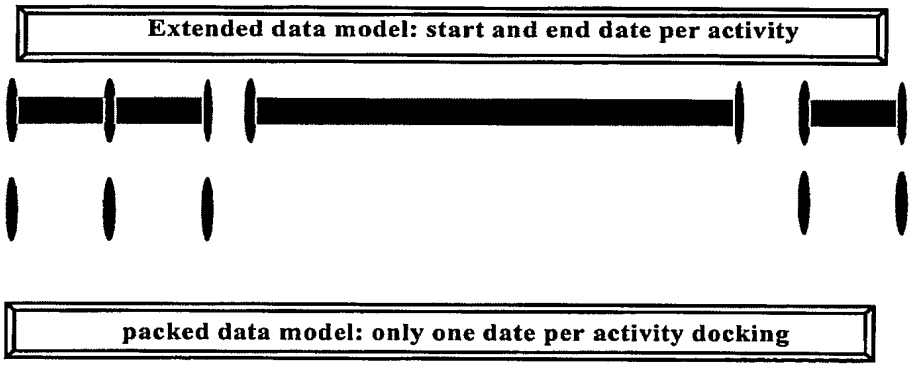



include a data model and maintenance tools for the definition of scheduling schemes. This will include function based tools which might be used by a business engineering tool which is used by SCFC for the definition of a complete business process. Thus scheduling modelling may be included in a more complex set-up of business processes.
In the figure a example is presented of a schedule scheme which consists of the scheme already used in existing SAP APO scheduling function and additional dates which may be used for more complex description of the picking activity and the transportation activity.
Please note that by defining complex scheduling schemes SC scheduling will be able to process any complex transport consisting of several route stages and load transfer points.

### 2.1.3 Scheduling result data models

Existing schedule functions of SAP R/3 and APO have different data designs. Schedule functions used by sales order processes have a data model where only one date per relation of activities is known. For example material availability date, loading date, goods issue date and delivery date. Those four dates describe four activities: picking, loading, transport, unload. This data model is in contrast to the data model used by planning applications ( SNP or TPVS in SAP APO ) which describe an activity by start-date and end-date.

Because SC basis scheduling will have to serve for both types of existing SAP applications it will support both data models. It will schedule the activities and provide details by a result data model like the planning applications already are used to and it will also provide the scheduling result in the more packed data model like sales order processes are used to. For the packed data model additional date types have to be defined. In case the conversion from the detailed data model to the packed data model is not straight forward, a set of rules for the transformation will be used. These transformation rules will be configurable and will be defined per scheduling scheme. The most simplest way of defining a transformation rule would be to set a date type of the packed format equal to the date type of the extended format. If one may need more complex rules is an open question ( see open items).
The package of SC basis scheduling will provide a data model and maintenance tools for the definition of these rules. This will include function based tools.
The interface of SC basis scheduling is in terms of time zone UTC. The result of the scheduling, the scheduled dates, will also be in terms of time zone UTC. The schedule result will contain for each date a time zone which might be used for presenting the date at the GUI. Note that this time zone will be initial(UTC) if the time zone determination fails due to missing master data.
The picture below indicates the difference between the extended data model ( as used by SAP APO SNP ) and the packed data model ( as used by SAP sales order ). For example, at the connection of activity 3 and 4 ( loading and transportation ) in the packed data model only one date is known. This so called goods issue date is the end date of activity 3 (loading).



## 2.1.4 Options for control of scheduling algorithm

### 2.1.4.1 Constrain earliest timestamp

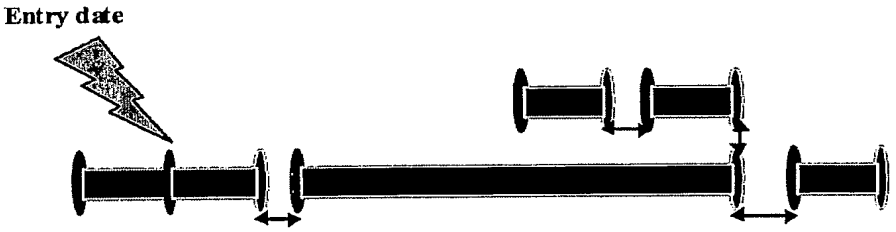
The interface of the scheduling service contains a date(timestamp) which is used by the scheduling algorithm as a constraint on the earliest allowed date. If this option is used, no date before this specified earliest date will be contained in the result of the scheduling.
The constraining earliest timestamp will have a optional additional specification which specifies the dates for which the constraint should be applied. In case a date of the packed data format is used in this specification, a conversion to a date of the extended data format is done. Thus in the definition of a scheduling scheme the transformation rules between extended and packed data format has to include both directions of transformation.

### 2.1.4.2 Scheduling scheme

The interface of the scheduling service contains the scheduling scheme which shall be used for the scheduling service call.

### 2.1.4.3 Entry point date

A entry point date has to be specified in the interface of the scheduling service. This includes a timestamp ( in UTC ) and a date type. The scheduling algorithm will use this date as a starting point. Any date type is allowed as entry point, as long as it is a valid date type of the scheduling according to the scheduling date type definition ( see section definition of activities ) and as long as it is contained in the scheduling scheme which is used.
In case a date of the packed data format is used as entry point, a conversion to a date of the extended data format is done. Thus in the definition of a scheduling scheme the transformation rules between extended and packed data format has to include both directions of transformation.



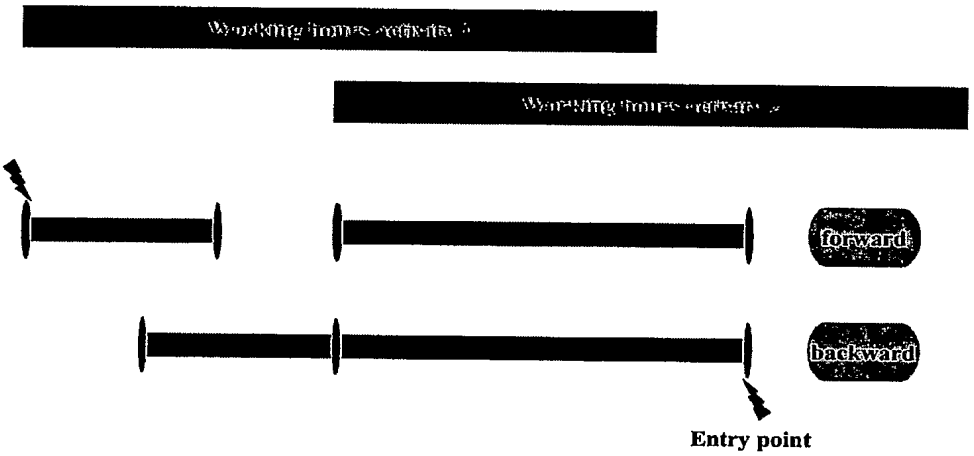

### 2.1.4.4 Additional entry point dates

In general, the scheduling algorithm may not be reversible. For example a scheduling with entry point delivery date LFDAT_1 results in a material availability date MBDAT. A subsequent scheduling with

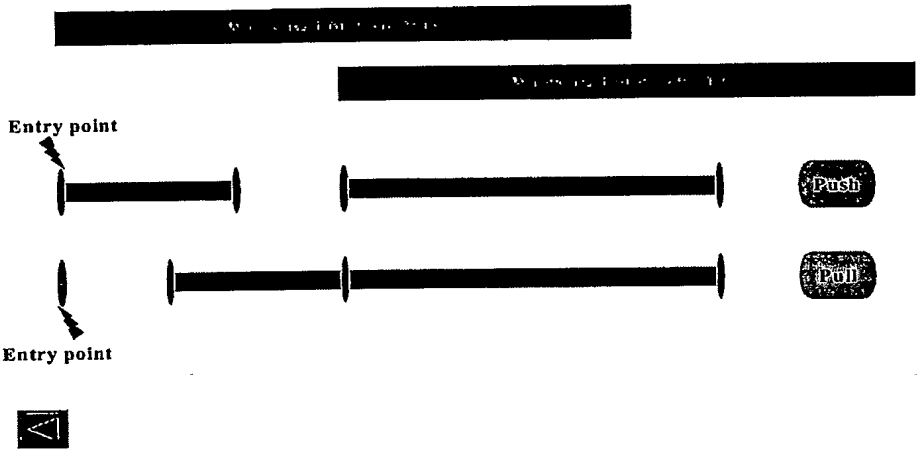
entry point MBDAT result in a delivery date LFDAT_2 , where LFDAT_2 is not equal to LFDAT_1. The fact that LFDAT_2 is not equal to LFDAT_1 is called non-reversible in this specification.
Because some applications rely on the reversibility of the scheduling algorithm, the interface of the scheduling service contains an optional table of additional entry points. If this table is given by the caller application, these additional entry points will be used first ( in sequence ) for a scheduling. In case of that the result of one of these scheduling meets the main entry point, no further scheduling is processed and the result of this individual scheduling is the result of the scheduling service call.
Thus, in case of the example above, one may perform a roundtrip scheduling
LFDAT->MBDAT->LFDAT
by a first scheduling service call with entry point LFDAT ( and no additional dates ) which calculates MBDAT and a second scheduling service call with entry point MBDAT and the additional entry point LFDAT. Note that in this roundtrip scheduling the non-reversibility is not seen by the caller application because the roundtrip is emulated within the scheduling service rather than really performing it.

### 2.1.4.5 Push or pull optimisation

The SAP R/3 and SAP APO transportation and shipment scheduling algorithms perform ( only to some




extend ) a pull optimisation. This means, that the scheduling algorithm optimises the scheduling result such that the material availability date is as late as possible ( without changing the delivery date ). Thus, material procurement has to be made as late as possible. In case of a pull business process ( like customer sales order, VMI order , ... ) this optimisation is likely to improve business profits.
In case of push processes one may optimise the material availability date to be as soon as possible. Examples of push processes are the material distribution from a production plant ( with limited storage capacity ) to distribution centers which have larger storage capacity and perform a 'sales from stock' business.




The SC scheduling interface will contain one date type which denotes the date which shall be fixed in this optimisation ( please note that this does not guarantee that this date remains unchanged in the whole scheduling ! ) For example, in case of sales order processing this can be the delivery date, because optimisation should not influence the confirmed delivery date. In addition the scheduling interface contains a push or pull parameter which specifies which optimisation method should be applied. In case of pull, the optimisation will optimise all dates which are according to the scheduling scheme before the fixed date towards later dates. In case of push, the optimisation will optimise all dates which are after the fixed date towards earlier dates. All other dates are not contained in the scope of the optimisation. In case of complex scheduling schemes the restriction applies that only those dates are optimised which can be accessed starting from the fixed date by a propagation with one direction in time only. Thus the dates of parallel activities might not be optimised. Example: in the figure which shows the complex schedule scheme in chapter 4.2.2 the fixed date might be 'start of activity 4'. In case of pull optimisation activities 1,2,3,6,7 will be optimised towards later dates. In case of push optimisation the other activities will be optimised towards earlier dates.
The figure shows an example where the end of activity 2 shall be the fixed date within the optimisation. In case of push, no other activities after this date exist, thus optimisation is trivial ( no work to be done at all). In case of pull both activities 1 and 2 are before this fixed date, thus optimisation takes place. Technically the optimisation will be done by a backward scheduling starting from the fixed date.

### 2.1.4.6 Maximum number of internal and external iterations

The SC scheduling service has a optional parameter to set the maximum number of internal and external iterations. An internal iteration is defined to be the calculation of a scheduling activity by SC basis scheduling. An external iteration is the call to a external scheduling method or a call to a external check method ( see chapters below ).
By means of those two parameters the maximum runtime of a single SC basis scheduling service call can be constraint. This is necessary due to the fact that the SC basis scheduling algorithm is in general not a serial calculation. Instead it contains a iteration feature. This iteration feature takes place in case of optimisation and the usage of methods for availability check. In general, this iteration can have a non-converging behaviour !
The maximum number of iterations defines a exception. In case of the exception, no scheduling result will be available. In order to get a scheduling result, the scheduling has to end the non-converging iteration in a well defined manner. This might be done by changing the parameters of the scheduling

request such that a converging iteration is obtained. The result of this converging iteration can then be used as the best-obtainable result of the original scheduling request.

#### 2.1.4.7 Convergence strategy

In order to overcome possible problems due to non-convergence behaviour of a given scheduling request, the SC basis scheduling service contains a optional parameter which enables to set a strategy for transfer of the non-converging problem into a converging iteration. For example, this strategy may stop the push/pull optimisation in case a non-convergence is observed ( detected by a maximum number of allowed internal iterations). By usage of the convergence strategy the scheduling service will try to determine the best-obtainable result internally within the service call. Thus the calling application will not have the need to implement it's own re-trying of the scheduling service with different options.
Further details on the convergence strategy needed are not known so far. But this lack of knowledge can be fixed by means of a prototype of the scheduling algorithm and investigations on the iterations of this prototype.

#### 2.1.4.8 Time unit used by SC basis scheduling

By default SC basis scheduling will schedule with a precession of a second. Some applications may like to schedule with an other precession like minutes, hours or days. SC basis scheduling will provide the possibility to select the precession by specifying a time unit. Allowed time units are

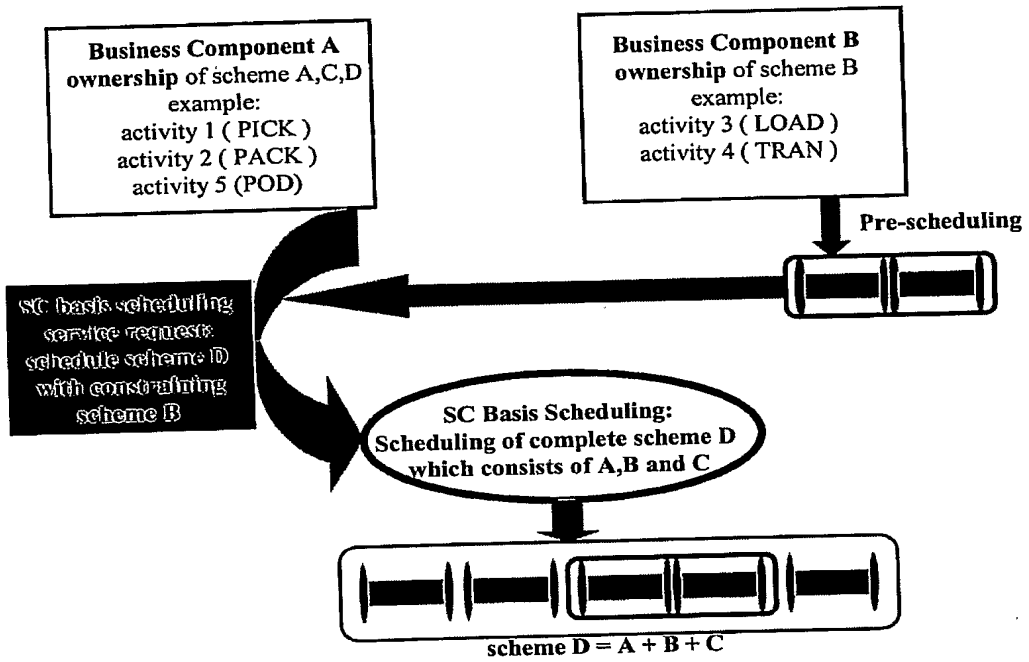
- second
- minute
- hour
- day

SC basis scheduling will use these time units as a precession unit for the internal algorithm ( scheduling of individual activities ) as well as for the interface data. For all time units different to second the middle of the time bucket will be used in those cases when a timestamp with precession of a second has to be maintained. ( high noon in case of a day ). In the scheduling individual activities contains data with a precession of a second ( working hours, duration ), then first a scheduling of an individual activity with the precession of a second is performed ( by means of time stream basis package ) and then the result is rounded to the required precession. This rounded result is then used for further scheduling of other activities. Thus SC basis scheduling will not apply a rounding to the master data. But obviously the SC basis scheduling will be most stable with master data which has same precession as the required time unit.

### 2.1.5 Scheduling in enhance mode with constraining fixed dates

Business processes handled by SCM will make use of several applications and services of various sub-systems or components. If one of these components has the ownership of a part of the logistic dates there is a need for fixed activities within the SC basis scheduling service. This leads to a constraint based scheduling. One may define not only fixed activities but may also extend it to defining fixed scheduling schemes ( which are a sub-scheme of the actual scheme which is used by SC basis scheduling ) or use a single constraining date.

Fixed activities/sub-schemes have to be pre-set ( scheduled ) before they can be used within SC basis service call. The dates of the fixed activities/sub-schemes have to be provided by the caller application via interface of the SC basis scheduling.




The figure shows an example where two different components A and B have the ownership of two scheduling schemes A,C,D and B. Scheduling schemes A,B and C can be combined to an overall scheduling scheme D. In the example, component B first performs a scheduling for scheme B ( without knowledge of component A ). Component A has the business needs to perform an overall combined scheduling of the complete business process, which is scheme D. This is performed by a SC basis service request which includes the pre-set scheme B as a fixed constraining scheme in the interface data of the SC basis service call. The calling application of component A has to retrieve the data of the constraining scheme B from component A.

Primarily, scheduling in enhance mode is intended to be used for collaboration business scenarios. An external business partner may describe his part of the logistic chain not only by one single date ( like requested delivery date ) but by a complete scheme of dates. This allows for scheduling additional dates ( which depend on the dates of the externally provided scheme ). In the example above business partner A (manufacturer) collaborates with an external logistic provider who does the transportation. The manufacturer may be interested in details of the time schedule of the transport. For example, he may use these details for defining certain deadlines for tracking. In addition, the manufacturer may have activities to be done in sequence to the transportation and unloading, for example processing a proof of delivery given by the goods recipient.

Scheduling in enhance mode may also be used by a transportation provider who plans transports by using SCM. SCM transportation planning applications might receive the constraining dates from manufacturer/goods recipient and perform a transportation scheduling by means of SC basis scheduling. Transportation may consist of several partial transports, for example truck → harbour → ship → harbour → truck. In case the transport is done by several different transportation providers, the overall planning might be done by iterations of SC basis scheduling with constraining sub-schemes which denote the part of the transport done by one of the transportation providers. The iteration process may be based on tendering the most critical parts of the transport first (in the example this is the ship ) . When the ship is

confirmed with certain dates, this might be used for a constrained based scheduling of the overall transport.
Please note the following features of scheduling with constraining fixed dates:

- it is convenient that the entrypoint date of the scheduling shall be one of the constraining fixed dates. If another date is used as entrypoint in the schedule of the complete scheme there might occure some time gaps. Example: transportation is a fixed constraining activity. Scheduling is done with entrypoint 'start of picking'. Thus picking and loading is scheduled, which may lead to a end-of-loading date several days before the fixed start-of-transport date.
- One may use a entrypoint date which is not one of the constraining fixed dates. In case of push/pull optimisation the optimisation algorithm might perform a additional scheduling of a part of the scheme which might resolve the time gap due to the inconvenient entrypoint date.

#### 2.1.5.1 Constraining dates: fixation categories

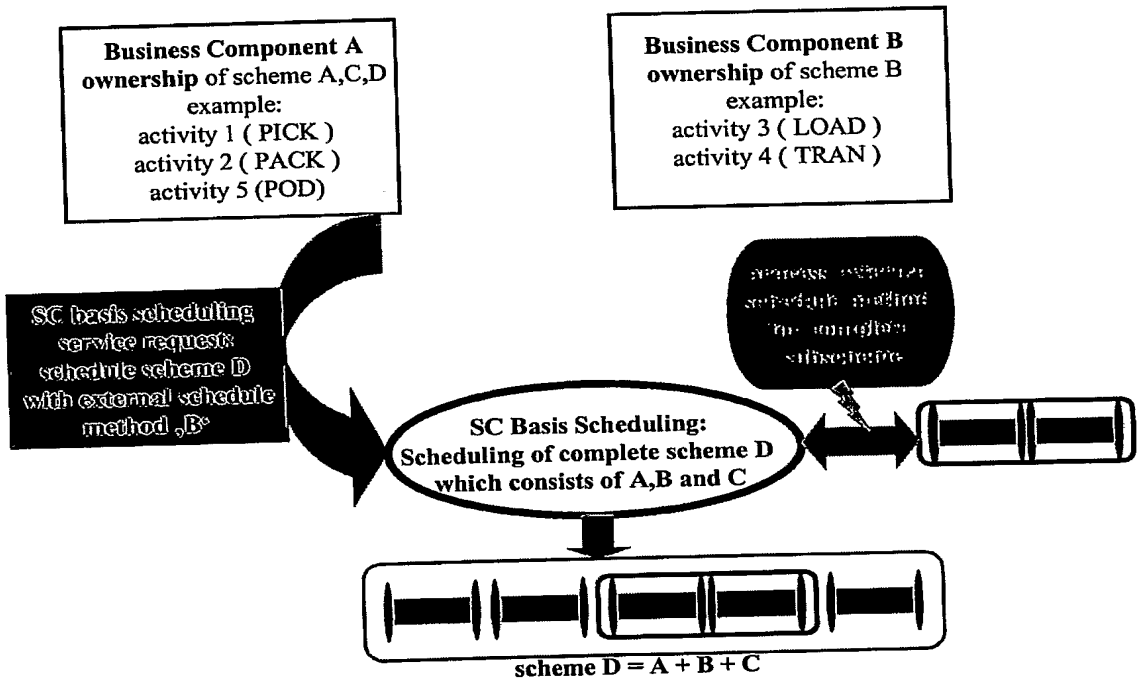
One may think of constraining dates which are not completely fixed. One may think of several categories of fixation:

1) slight fixation of constraining dates by means of an allowed time interval. In addition to the allowed time interval SC basis scheduling may need a step size which should be used for changing the dates within the scheduling algorithm.
2) horizon driven fixation. This means that only those constraining dates are fixed within the scheduling algorithm which are not very far in the feature. Constraining dates which are later than the specified horizon may be changed within the scheduling. Horizon driven fixation makes sense because the application does not have to maintain the horizon logic for it's application individually but uses instead the SC basis scheduling. The horizon may be specific for each activity of a business process, because some activities may have the need to be fixed very earlier. ( For example the activity which describes the transport by means of a large ship ).

### 2.1.6 Scheduling within a multi-component framework

SC basis scheduling will allow for a cross-system(cross-component) scheduling all in one service call. This means that certain activities or sub-schemes may be owned by remote systems ( or other components placed on the same system ) and that SC basis scheduling will schedule these activities/sub-schemes by usage of services of the other components which are the owner of this part of the business process. From point of view of SC basis scheduling these activities/sub-schemes will be used analogous to the constraining fixed dates within one iteration. Note that in case several iterations are necessary ( for example due to push/pull optimisation ) several calls to external scheduling services will be needed.
At first place, SC basis scheduling will be able to use synchronous external scheduling services. At a later stage of development one may include also asynchronous services. The package of SC basis scheduling will provide a data model and maintenance tools for the definition of scheduling methods available at SCM which shall be used for scheduling of sub-schemes. The subscribing of scheduling methods to activities/sub-schemes will not be static. It will be dynamically depending on application data which defines whether and which methods shall be used.
The scheduling methods have to be defined at SCM locally. If necessary, these methods will do a data mapping and remote function call. This may be done by usage of application integration tools of SAP basis ( like AI ).
These scheduling methods will not belong to the package of SC basis scheduling. They have to be provided by the applications that make use of the SC basis scheduling. The package of SC basis scheduling will contain a interface definition of these scheduling methods. This will include a generic data container which can be used for routing input data to the various scheduling methods through the SC basis scheduling. Thus SC basis scheduling can be adapted to any needs of application defined scheduling methods.
In addition, the concept of scheduling within a multi-component framework provides flexible user exits which allow for modifying the standard SC scheduling for certain activities/schemes in specific business

processes. In order to do so, one has to define customer specific scheduling methods which have the standardised interface implemented.
A example for a external scheduling service is the usage of the SAP APO transportation and shipment scheduling.
The figure shows the same example described in the previous section. Now the example does not use a constraining sub-schemes but executes an external schedule method each time the sub-scheme has to be calculated within a scheduling iteration step.

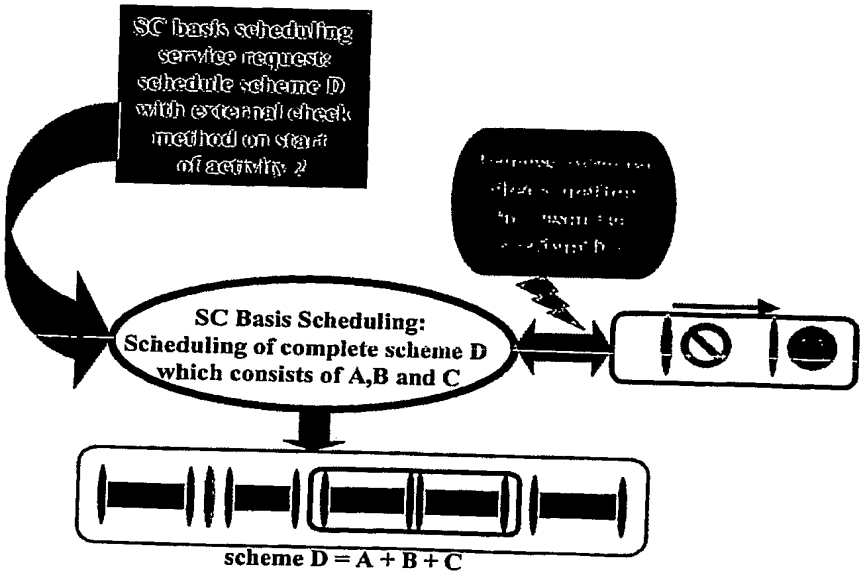



SC Basis scheduling will keep the information on the very last call for each external scheduling method for each activity in a buffer. This will include the interface data which was used for the call of the external method. The result interface of SC basis scheduling will contain a reference to this buffer. Thus, applications which use SC basis scheduling are able for doing some additional work connected with the external schedule method. ( for example see next chapter: a schedule method which is used as a check method against resource capacities over the whole time span of an activity period may have the need to allocate these resources when saving an order ).

### 2.1.7 Scheduling with finite resources

For each activity of a business process one or several resources may be needed. One example is that you need a truck and a truck driver for transportation. Scheduling with finite resources implies a check on the availability of resources needed by the activities. As an example, this check on available resources may include an ATP check for material availability date, a check on transportation resources ( for example performed by a quotation check if it's an external logistic partner who does the transportation ). For each scheduling activity one single check on finite resources can be performed on the start date of the activity. This check may change the start date of the activity towards a later date. Please note that in case of a scheduling iteration which performs a backward scheduling this date shift will lead to an additional iteration because the previously determined dates ( which are after the delayed start date of the activity ) are no more valid.

The external check method may indicate to the SC basis scheduling that the confirmed date and any later point in time is a valid start date of the activity. Thus, SC basis scheduling will be prevented from multiple calls to the external check method in case several iterations within the scheduling are performed. Please note, that SC basis scheduling does not have any knowledge of master data used within the check method. This includes also resource working hours which may not be identical to the working hours of the scheduling activity.
In addition, one may think of a check on the resource capacity over the whole time span where a resource is needed ( the duration of the activity ). This second type of check may delay the start and end date of the activity. This second check method is equal to applying an external schedule method for the activity. Thus, in SC basis scheduling the purpose of capacity check will be done by usage of external schedule methods.
The subscribing of check methods to activities/sub-schemes will not be static. It will be dynamically depending on application data which defines whether and which methods are used.
SC basis scheduling will not provide check methods. SC basis scheduling will provide an interface definition for the check methods. This will include a generic data container which can be used for routing input data to the various check methods through the SC basis scheduling. Thus SC basis scheduling can be adapted to any needs of application defined check methods. The check methods which subscribe to activities have to be available on the local SCM system. At first place, synchronous check services will be used. At a later stage of development one may include also asynchronous services.
Check methods will receive a required quantity of resource in base units and a point in time as input parameter ( which was determined by scheduling calculations). The check methods result have to consist of a confirmed quantity ( against requested quantity ), and a confirmed start-date of the activity. The check result may consist of several partial confirmations. The partial confirmations will be scheduled by SC scheduling independently. Please note that in case of partial confirmations the dates determined so far may have to be re-scheduled if scheduling master data depends on the quantity ( for example if activity duration depends on the weight group ).
Partial confirmations will lead to several sets of confirmed dates of the overall scheduling scheme, which is the result of SC basis service.
In case of checks on the resource capacity over the whole time span where a resource is needed ( the duration of the activity ), partial confirmations are not possible ! External schedule methods ( which are used for this kind of check ) will not have the option of splitting up the business process.



Check methods will be allowed for confirmation by substitution ( like GATP in SAP APO ). The substitution will be allowed for any application attribute, it will not be limited to shipping location/product as in GATP of SAP APO. Check methods will have to indicate substitutions in the interface data which is received by the calling SC basis scheduling service. In case of substitution SC basis scheduling will have to re-schedule those dates that have already been scheduled by using the substituted application data. In order to overcome possible endless loops the scheduling algorithm has to use predefined convergence strategies. Please note that substitution is not allowed for capacity checks which are performed by applying a external scheduling method.
The figure shows the same example described in the previous section. Now the example does use a check method to check the resource availability at start date of activity 2. In the figure, the start of activity 2 is delayed due to missing availability.
SC Basis scheduling will keep the information on the very last call for each external check method for each activity in a buffer. This will include the interface data which was used for the call of the external method. The result interface of SC basis scheduling will contain a reference to this buffer. Thus, applications which use SC basis scheduling are able for doing some additional work connected with the external check method.

#### 2.1.7.1 Route schedule

In SAP R/3 and SAP APO the shipping and transportation scheduling may use departure dates ( given by route schedule master data or set within a user exit ) to constrain the start of the transportation activity. Because these functions provide the scheduling result in terms of the packed data format, instead of the start of the transport activity the end of the loading activity (post goods issue date ) is aligned according to the constraint.
In SC basis scheduling, constrain on departure may be implemented by applying a external check method. This external check method may change the start date of an activity. In contrast to the external check methods discussed so far, the check on departure date constraints may change the start date not only towards a later point in time but may also change it to an earlier point in time. Thus the import interface definition of the external check methods has to include a flag which indicates the time direction (backward or forward ) which should be used to determine the next possible date. In the example of a classical backward scheduling ( starting from requested delivery date ) this will ensure that a requested delivery date can be met by changing the start date of transportation towards a earlier point in time ( which is according to the constraining departure dates ).
In case classical check methods ( like ATP check ) are called with backward scheduling indicator, the indicator will not change the result of the check method. These check methods will only confirm a later point in time. Thus, the time direction indicator denotes the time direction which should be used first in order to try to get a confirmed date. If this first direction fails, the other direction may be used also for confirmation.

### 2.1.8 Internal determination of master data or input via interface

In general, all master data used by SC scheduling can be determined within the scheduling or can be pre-set by the calling application via interface of the scheduling service. Note that the internal determination is limited by the amount of pre-set data via interface.
The master data we are talking about here include not only classical master data like location data or material data but extends also to the definition of the scheduling procedure itself, like the scheduling scheme, scheduling sub-schemes and the external methods for scheduling and availability check that shall apply for certain schedule steps. Thus SC basis scheduling will provide the possibility to make use of externally defined business processes ( for example business process definition by SCFC ). This external definition of the business process can be submitted to the scheduling via interface of the service for a single scheduling service call. Please note that this extended interface of the scheduling service will allow for a scheduling of a scheme which was not defined before my means of scheduling master data maintenance. This might be used for ad-hoc business processes that have to be scheduled. For regular repeating business processes a definition of a scheme by scheduling master data maintenance will be the adequate method. In this case the scheduling service will be called by the scheme identifier only.

The SC basis scheduling will make use of master data located at the local SCM system where the scheduling is processed. This master data will be used to determine which duration, working time calendar(time stream) and time zone have to be used for scheduling an activity. SC basis scheduling will provide a data model and maintenance tools for the definition of the evaluation procedure of duration, calendars and time zones which should be used in a individual scheduling calculation. This evaluation will dynamically depend on application data, probably making use of the condition technique.
The master data of duration may be set depending on the start or end date of an activity. Thus a 'rush hour' may be implemented where the duration is longer than normal. Analogous to the weight group/volume group definition ( existing in SAP R/3 and SAP APO scheduling ) time periods may be defined. The duration may then be maintained depending on these time periods. SC scheduling calculation algorithm will choose the duration which is appropriate for the beginning of the propagation of the activity. Given that the propagation can be forward or backward, the algorithm will sometimes choose the duration valid for the start date of the activity and sometimes it will choose the duration valid for the end time of the activity. This will give additional reason for non-reversibility of the scheduling.
In addition, this data ( duration, calendar, time zone ) might be pre-set by the caller application for an individual scheduling service call, thus inhibiting the standard evaluation procedure within the SC basis scheduling. SC basis scheduling may be used for scheduling parts of business processes which are owned by other components by using pre-set master data of these other components. Thus SC basis scheduling can be used as calculation engine, without any master data determination within SC scheduling service. A example would be to extract scheduling master data first from SAP APO system and then perform a SC basis scheduling by pre-setting the duration, working calendars with the APO master data. This would allow for a SC basis scheduling which would be consistent to the scheduling of other business processes that make use of SAP APO scheduling only. Note that this will provide only a loose and not precise integration between the other components ( like SAP APO ) and SC basis scheduling because different scheduling algorithms are used. Equal master data is not sufficient for equal results. This can only be provided by usage of the constraint scheduling or usage of external schedule services.

### 2.1.9 Assignment of locations to activities

In many cases it will be convenient that SC basis scheduling gets the information on the location ID's of start and end of an activity. These location ID's might by given directly be the calling application or the SC scheduling master data evaluation will try to determine it ( by usage of customizing/master data and the application data given by the single scheduling service call ). The location ID might be used in subsequent master data evaluation where duration, working hours, external methods are determined.
Compared to SAP APO transportation and shipment scheduling, which uses only one pair of location ( ship-from and ship-to ) SC scheduling will use for each activity individually a pair of locations.
Thus, SC basis scheduling will be able to process business processes with a complex transport consisting of several partial transports. This includes for example transport from shipping location to customer A, from there to Customer B, from there to Customer C, from there backwards to the shipping location.

### 2.1.10 Scheduling explanation service

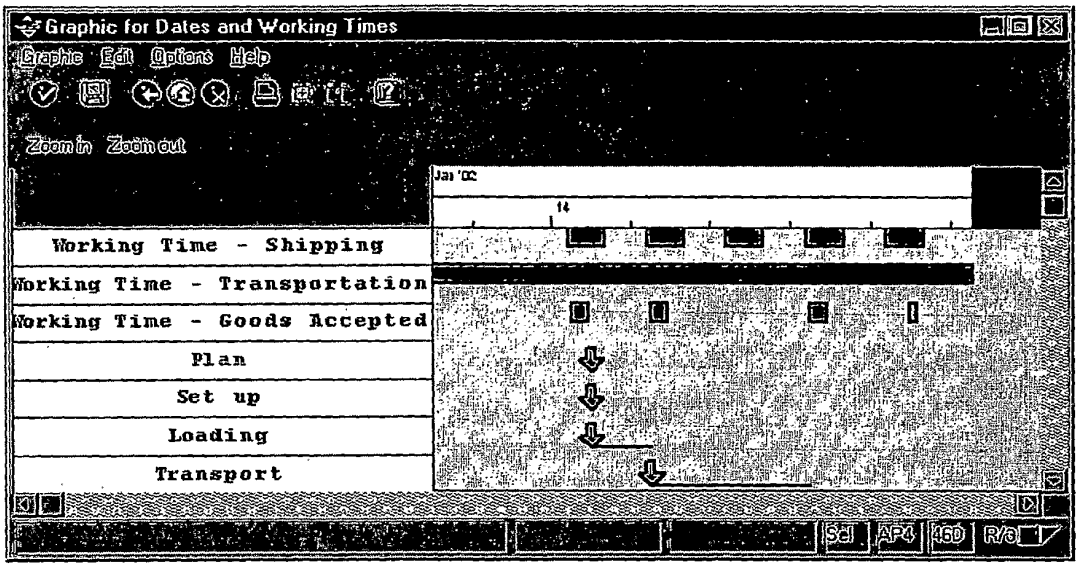
SC basis scheduling will provide services which allow for detailed analysis of the scheduling and thus preventing from a 'black box engine'. This includes

1) which working hours/duration have been used. This includes information on how this data was determined ( for example which master data was used or a log of the condition technique )
2) a protocol of the iterations processed within the scheduling algorithm.

3) A graphical display of the scheduling result and the constraints ( working hours, earliest allowed date , constraining dates , etc.)

This explanation service will be made available online by an optional parameter of the SC basis scheduling service. The explanation service will be available also offline by means of a SC scheduling log. Each time a schedule service is performed, a log can be written to data base. The SC basis scheduling will provide a GUID in the exporting interface which identifies the log belonging to this service call.

As an example on how the graphical display of the scheduling result may look like the picture below presents the SAP APO transportation and shipment scheduling explanation graphics. It is an example which contains for activities, but only two of them have a duration.

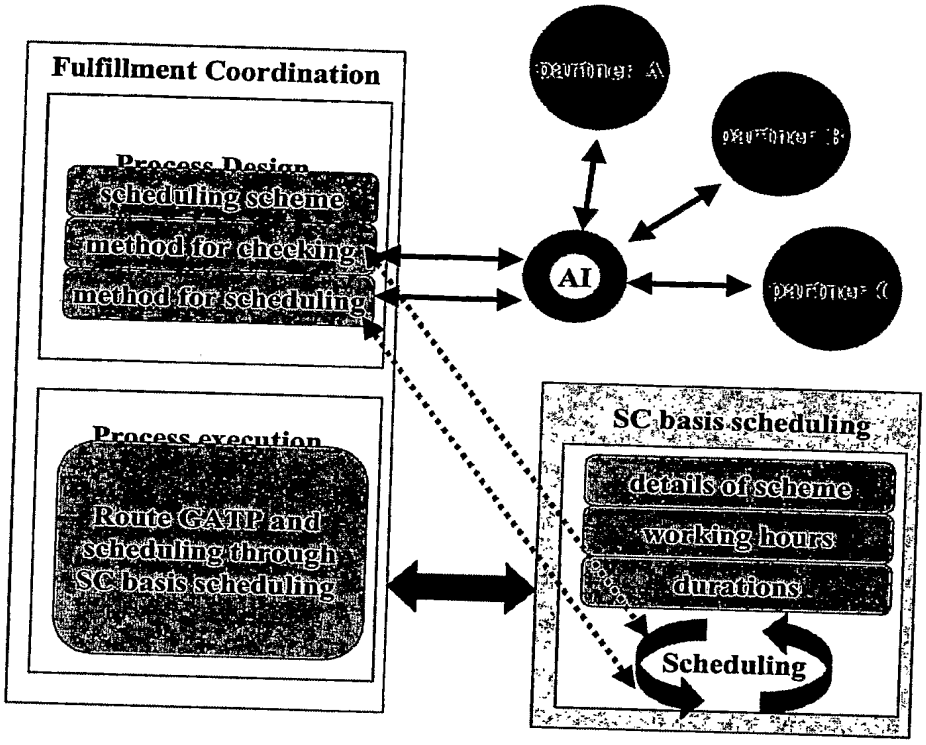



## 2.1.11 XML Service interface of SC Scheduling

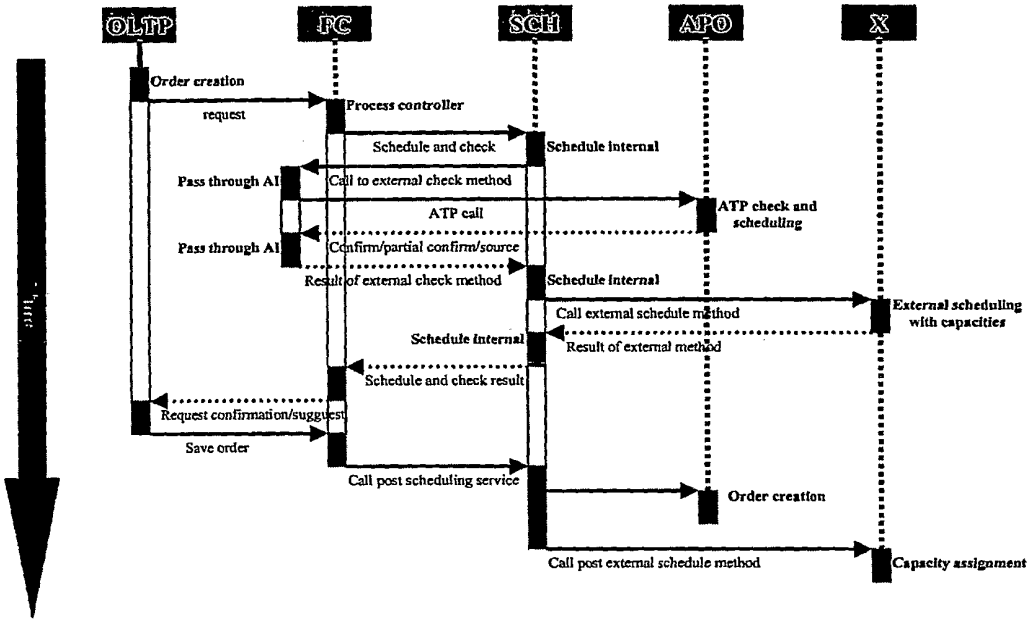
SC basis scheduling will have it's own XML interface which will allow for usage of the scheduling independently of any other application. The scheduling explanation service will have a XML interface also which will provide details on the scheduling ( but without dialogue ! ).

## 2.1.12 Integration of SC basis Scheduling with SC Fulfillment coordination

SC Fulfilment Co-ordination will be responsible for business process design and business process execution. If SCFC likes to make use of SC basis scheduling, the business process design of SCFC may maintain also the master data of the scheduling, i.e. scheduling activities, scheduling scheme ( which may include external scheduling methods and check methods ) and master data for evaluation of working hours and duration. Therefore the package of SC basis scheduling will include interfaces for master data maintenance services.
SC Fulfilment may not maintain any master data of the scheduling before a scheduling service call. In this case, the data which defines scheduling scheme, activities, external methods have to be supplied via the interface of the service call.
Once SCFC makes use of SC basis scheduling, the scheduling may use the check methods and scheduling methods defined by SCFC. Those methods may use the new SAP basis application AI for integration with other systems/components/partners.
The figure shows ( at a high level ) the integration of SCFC with SC basis scheduling.




As an example the following figure shows an example process where an OLTP system ( like SAP CRM or SAP R/3 ) does a order creation with SCFC which makes use of SC basis scheduling. Please note that the components shown in the figure ( called FC,SCH,APO,X ) are functional components which may be placed at the same system so that no remote function calls are needed for communication between them.




## 2.2 Configurability / Personalization / Customizing

### 2.2.1 Customizing

The following maintenance tools have to be available in the Customizing

1) Definition of scheduling activities ( this includes date types, duration types )
2) Definition of scheduling scheme ( this is the list of activities and the links which relate them in time )
3) Definition of master data evaluation procedures
4) Definition of convergence strategies

This data shall be maintained before first use of SC basis scheduling

### 2.2.2 Configurability

The following data might be maintained by means of condition records

1) duration ( per duration type depending on application data ). Note that the duration is per duration type and thus might be used for more than one activity. Example: duration type 'LOADING' might be used for activity loading at the shipping location and might be re-used for activity 'LOADING2' at a load transfer point. Due to the fact, that at the load transfer point the used application data differs

from the application data used for the loading at the shipping point, the duration of activity 'LOADING2' might be different. This can be implemented by maintaining condition data sets for the duration type 'LOADING' which depend on the location where the loading is done. Given that the SC scheduling has the information on the locations where a activity is done, the duration can be determined depending on this location.

2) Working hours per duration type. Note that the working hours is per duration type and thus might be used for more than one activity. In the example above the shipping point and the load transfer point may have different working hours.
3) In addition, functions may be defined which shall be used for determination of duration/working hours. The question which master data has most priority (function or condition data set) will be answered by the settings of the definition of the master data evaluation procedure. The functions will have an interface consisting of the complete application data which is submitted to SC basis scheduling. These functions will provide access to SAP standard master data. The package of SC scheduling will include a set of functions that might be used for this.
4) These functions may also be used as USER-EXITS for determination of duration/working hours.
5) Definition of external check/schedule methods by means of condition technique. The methods may not only depend on the activity type but may differ for different locations. ( Because for each location different business partners may have the ownership of the process ).

2.1

SAP Aktiengesellschaft

June 30, 2003
S50315



## Claims

1. Method of configuring a business process for scheduling,
the business process comprising a plurality of activities, each activity comprising at least one of a start date type and a stop date type; the activities being in a time relationship to each other; wherein
the business process is freely configurable with respect to the plurality of activities and with respect to the time relationships of the activities to each other.

2. The method according to one of the preceding claims, wherein a technical ID is associated with an activity or with a date type.

3. The method according to one of the preceding claims, wherein a text is associated with an activity or with a date type, the text being descriptive for the activity or for the date type.

4. The method according to one of the preceding claims, wherein time units are assigned to specific date types, the time units being freely configurable for each date type.

5. The method according to one of the preceding claims, wherein an activity can be modeled as a plurality of sub-processes.

6. The method according to one of the preceding claims, wherein a sub-process comprise a plurality of activities.

7. The method according to one of the preceding claims, wherein a decision whether or not a delegation is invoked is during run-time of the scheduling.

8. The method according to one of the preceding claims, wherein said service functions being usable for determination of time zone, calendar and duration of an activity.

9. The method according to one of the preceding claims, wherein at least one service function is assigned to at least one activity, the service function being usable, during scheduling, for determining start date and/ or finish date of the at least one activity.

10. The method according to one of the preceding claims, wherein at least one delegation scheme is assigned to at least one activity, the delegation the service function being usable for invoking, during scheduling, an external application for determining start date and/ or finish date of the at least one activity.

11. The method according to one of the preceding claims, wherein the activities and their time relationship are representable as a network of nodes and edges, each node representing one of the plurality of activities, and each edge connecting a pair of nodes and representing a predecessor-successor relationship of the activities represented by the respective pair of nodes.

12. The method according to one of the preceding claims, wherein a scheduling scheme is produced based on the configured business process, whereby the scheduling scheme is a set of meta data descriptive of how the individual activities are to be processed within scheduling.

13. The method according to claim 1, wherein a scheduling scheme is associated to the business process, the scheduling scheme comprising configuration data to at least one of duration, calendar, and time zone.

14. The method according to one of the preceding claims, wherein a scheduling scheme is associated to the business process, the scheduling scheme comprising configuration data to at least one of service function, and delegation process model.

15. A method of scheduling a business process, whereby the business process is configured with the method preferably according to claim 1.

16. Method of configuring a production process for simulating,
the production process comprising a plurality of steps, each step comprising at least one of a start date type and a stop date type; the steps being in a time relationship to each other; wherein
the production process is freely configurable with respect to the plurality of steps and with respect to the time relationships of the steps to each other.

17. A computer system for performing the method according to one of the preceding claims.

18. A computer-readable storage medium comprising program code for performing the method according to one of claims 1 to 16, when loaded into a computer system.

**Abstract**

A method and system of configuring a business process for scheduling the business process comprising a plurality of activities, each activity comprising at least one of a start date type and a stop date type; the activities being in a time relationship to each other; wherein the business process is freely configurable with respect to the plurality of activities and with respect to the time relationships of the activities to each other.

**22 Blatt Figuren**

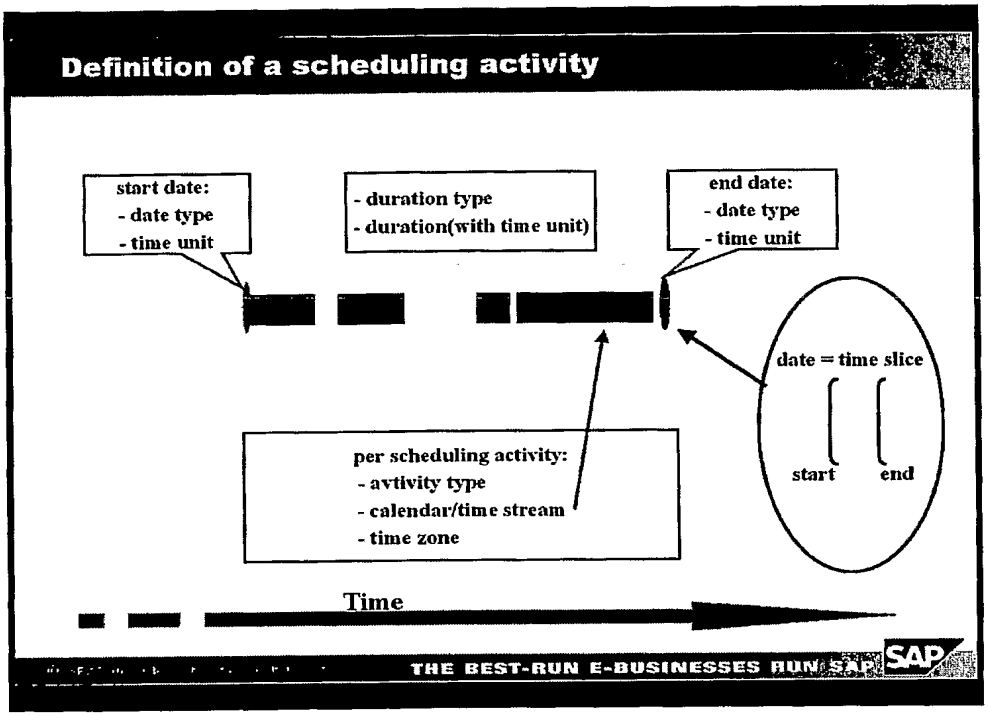



Fig. 1A

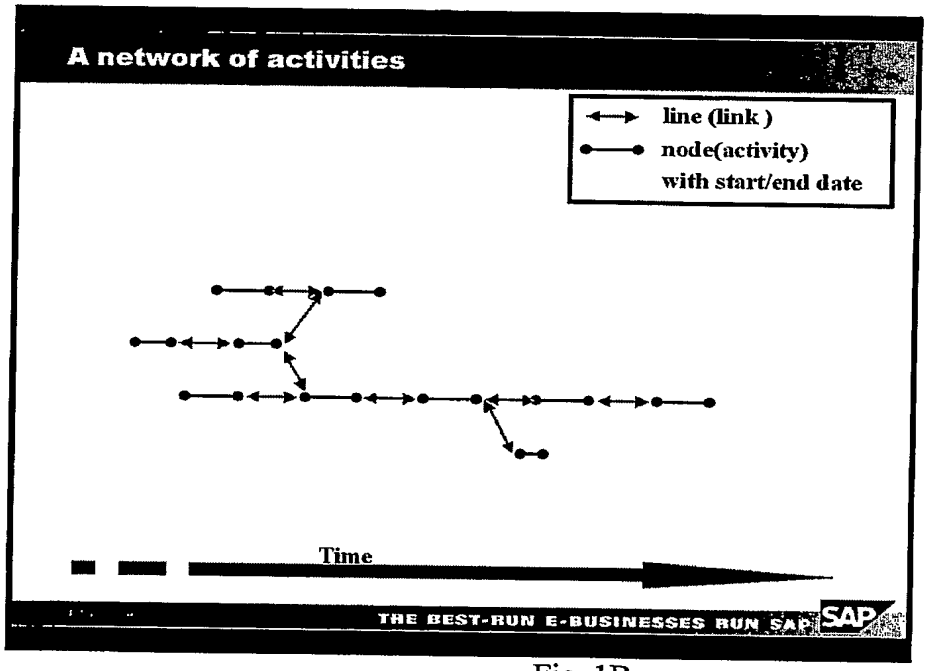

A network of activities
line (link )
node(activity)
with start/end date
Time
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 1B

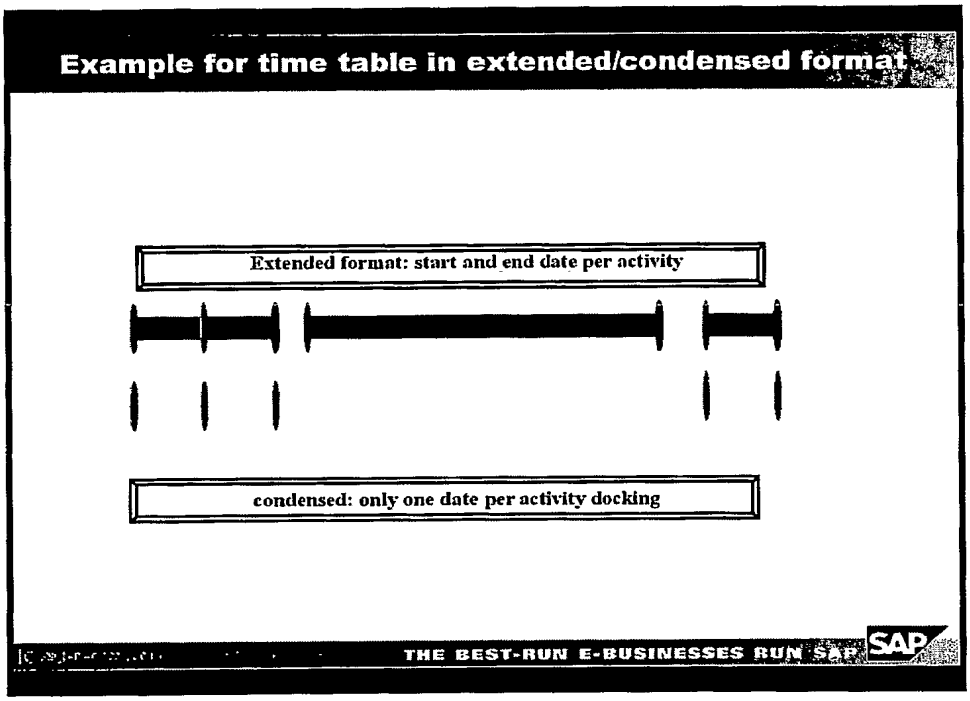

Example for time table in extended/condensed format
Extended format: start and end date per activity
condensed: only one date per activity docking
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 1C

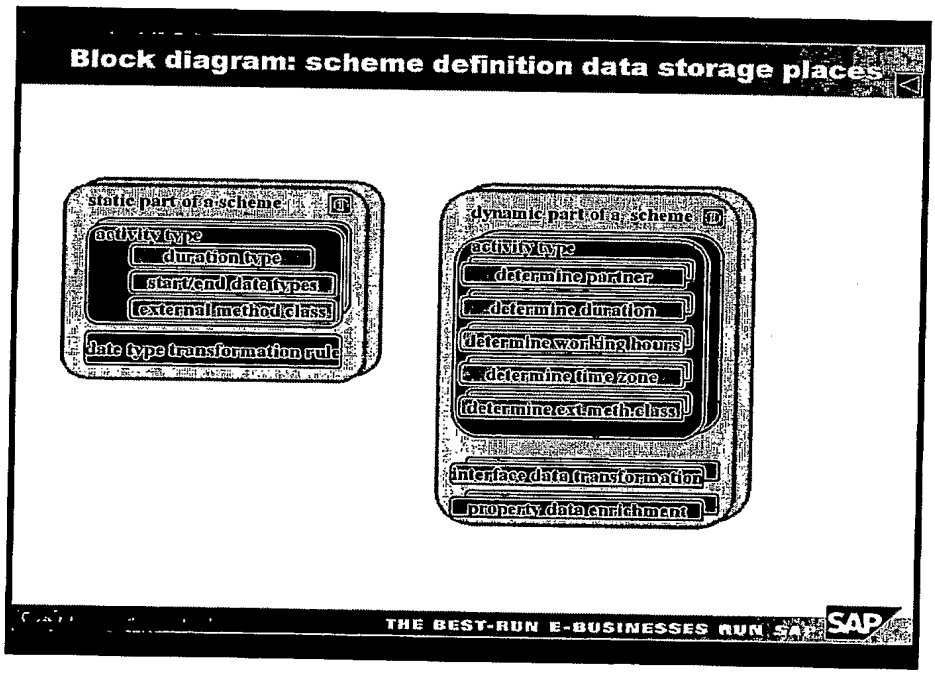
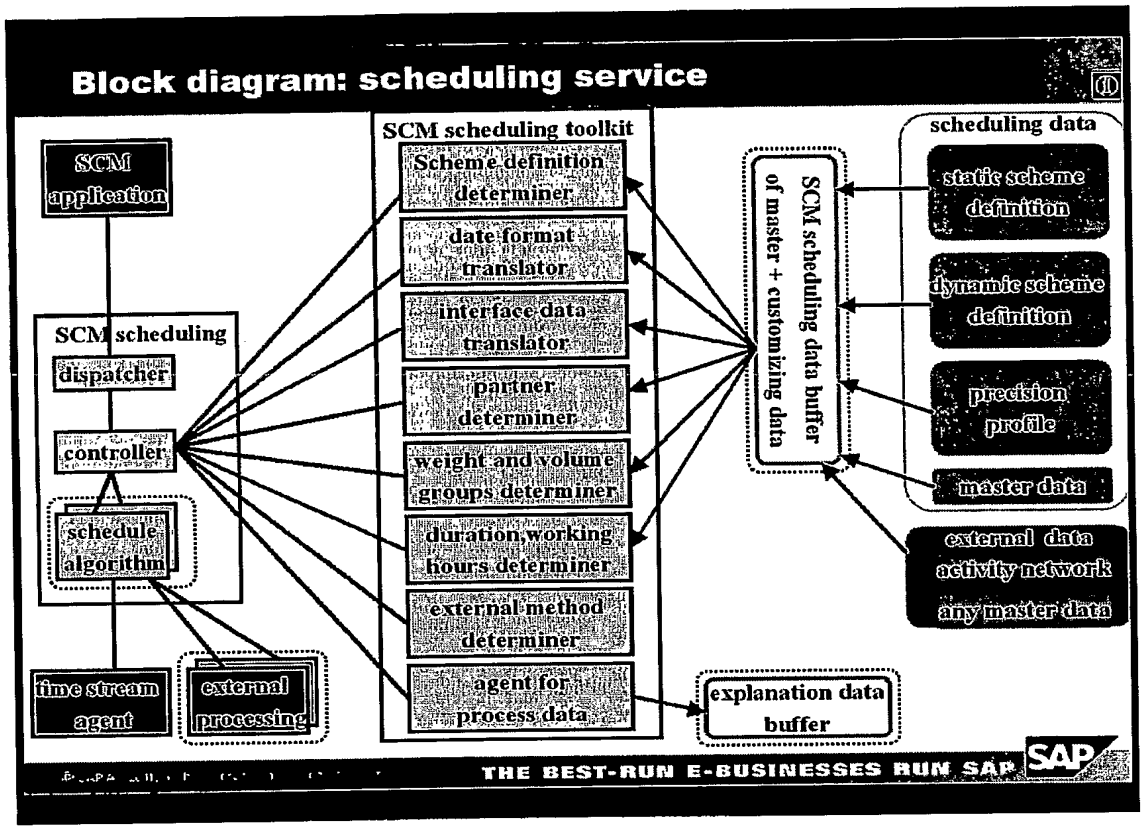

Block diagram: scheme definition data storage places
static part of a scheme
activity type
duration type
start/end date types
external method class
date type transformation rule
dynamic part of a scheme
activity type
determine partner
determine duration
determine working hours
determine time zone
determine ext meth class
interface data transformation
property data enrichment
THE BEST-RUN E-BUSINESSES RUN SAP

Fig. 2


Block diagram: scheduling service
SCM application
SCM scheduling
dispatcher
controller
schedule algorithm
time stream agent
external processing
SCM scheduling toolkit
Scheme definition determiner
date format translator
interface data translator
partner determiner
weight and volume groups determiner
duration working hours determiner
external method determiner
agent for process data
SCM scheduling data buffer of master + customizing data
explanation data buffer
scheduling data
static scheme definition
dynamic scheme definition
precision profile
master data
external data activity network
any master data
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig, 3

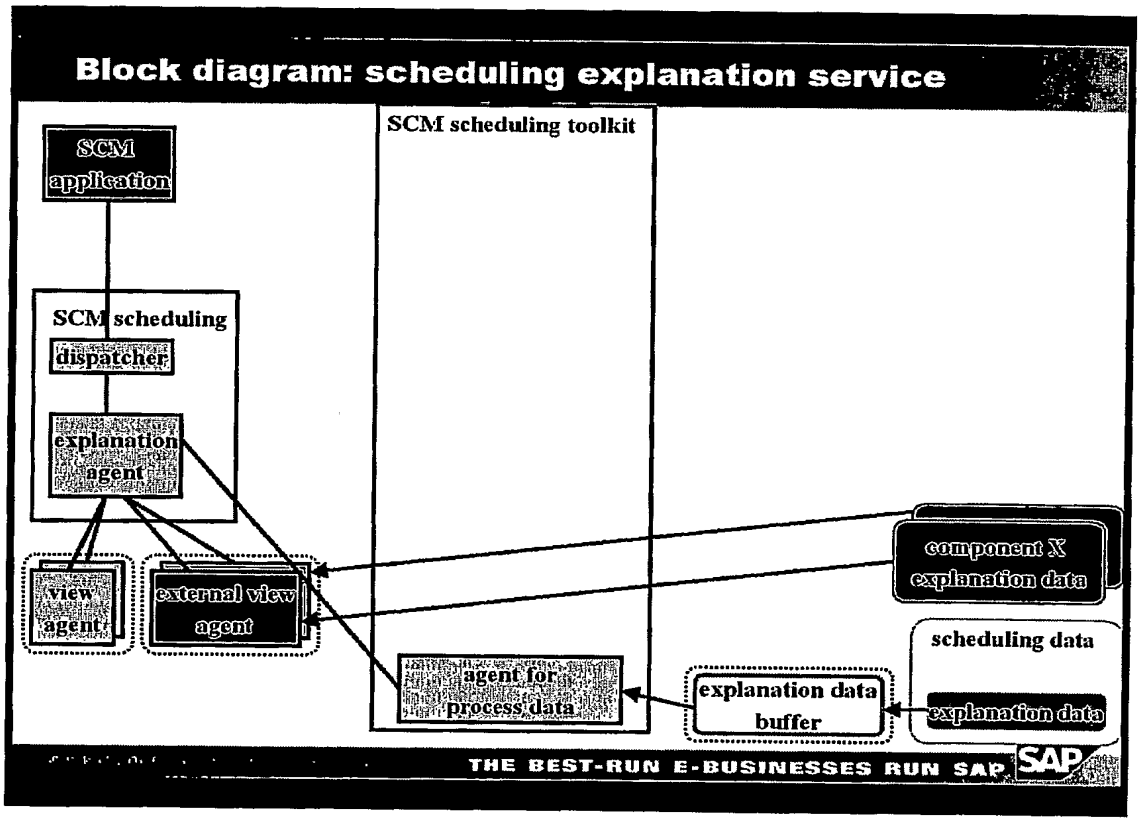

Block diagram: scheduling explanation service
SCM application
SCM scheduling toolkit
SCM scheduling
dispatcher
explanation agent
view agent
external view agent
component X explanation data
scheduling data
explanation data
agent for process data
explanation data buffer
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 4

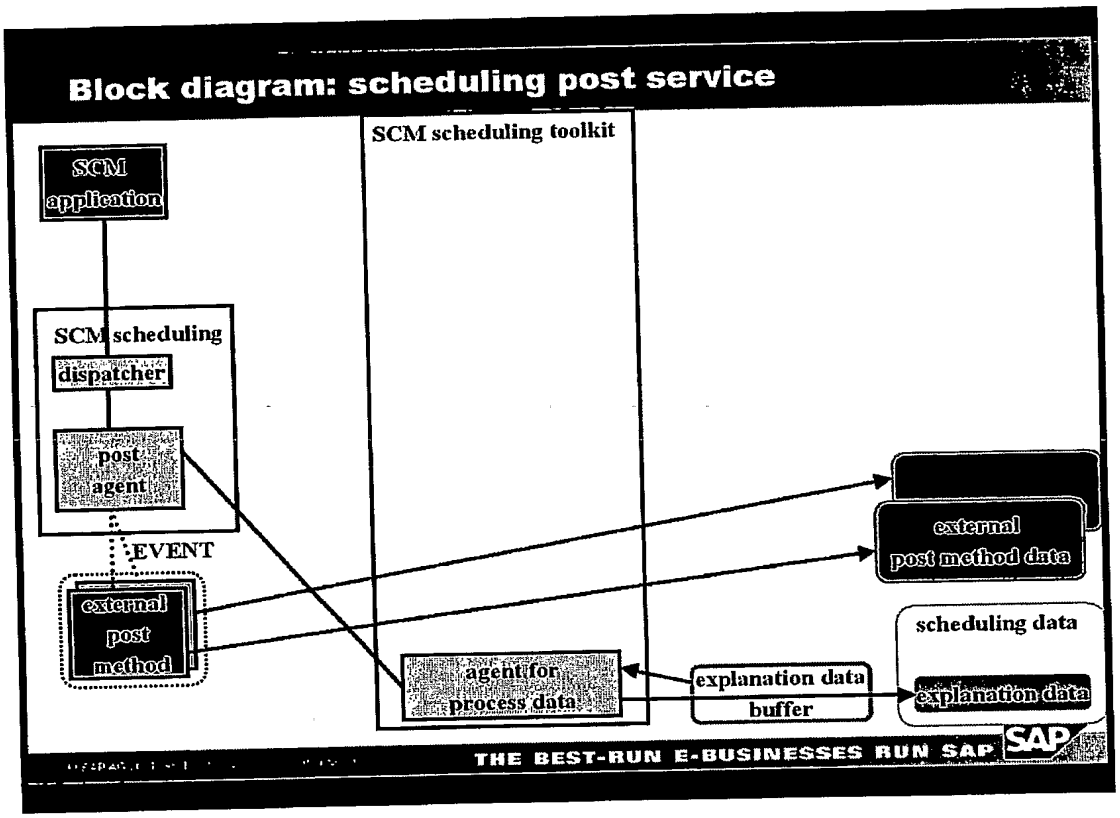

Block diagram: scheduling post service
SCM scheduling toolkit
SCM application
SCM scheduling dispatcher
post agent
EVENT
external post method
external post method data
scheduling data
agent for process data
explanation data buffer
explanation data
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 5

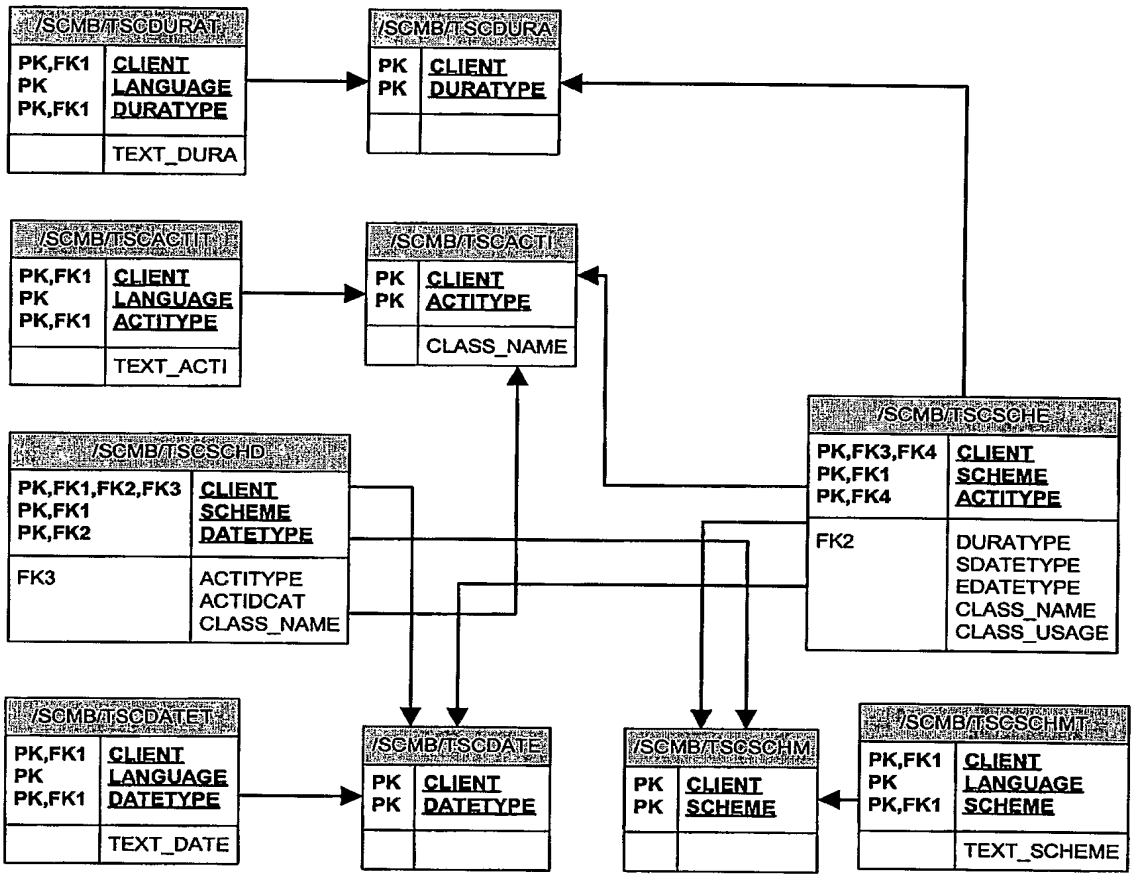

/SCMB/TSCDURAT
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 DURATYPE
TEXT_DURA
/SCMB/TSCDURA
PK CLIENT
PK DURATYPE
/SCMB/TSCACTIT
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 ACTITYPE
TEXT_ACTI
/SCMB/TSCACTI
PK CLIENT
PK ACTITYPE
CLASS_NAME
/SCMB/TSCSCHE
PK,FK3,FK4 CLIENT
PK,FK1 SCHEME
PK,FK4 ACTITYPE
FK2
DURATYPE
SDATETYPE
EDATETYPE
CLASS_NAME
CLASS_USAGE
/SCMB/TSCSCHD
PK,FK1,FK2,FK3 CLIENT
PK,FK1 SCHEME
PK,FK2 DATETYPE
FK3
ACTITYPE
ACTIDCAT
CLASS_NAME
/SCMB/TSCDATET
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 DATETYPE
TEXT_DATE
/SCMB/TSCDATE
PK CLIENT
PK DATETYPE
/SCMB/TSCSCHM
PK CLIENT
PK SCHEME
/SCMB/TSCSCHMT
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 SCHEME
TEXT_SCHEME

Fig. 6A

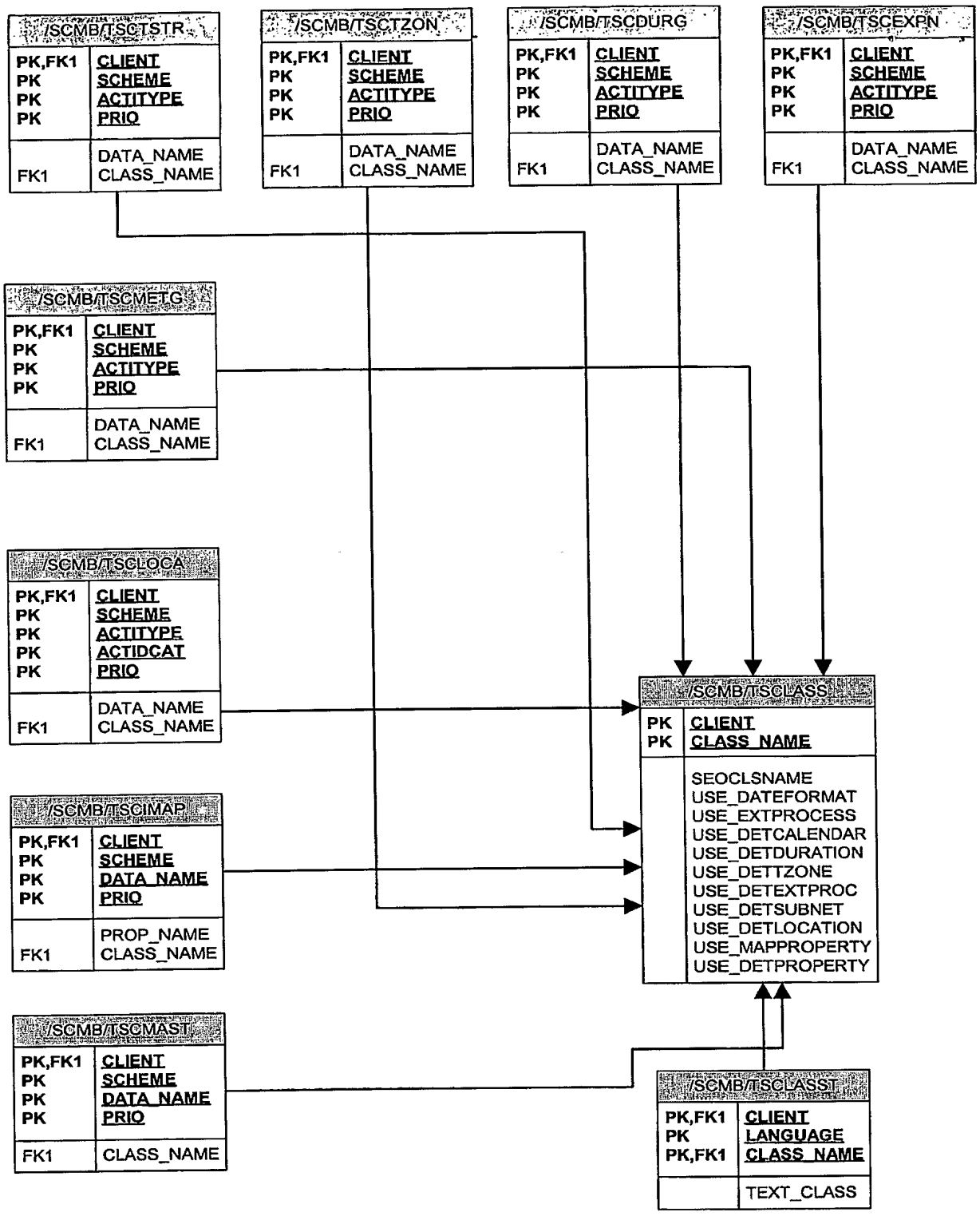

/SCMB/TSCTSTR
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCTZON
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCDURG
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCEXPN
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCMETG
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCLOCA
PK,FK1 CLIENT
PK SCHEME
PK ACTITYPE
PK ACTIDCAT
PK PRIO
DATA_NAME
FK1 CLASS_NAME
/SCMB/TSCLASS
PK CLIENT
PK CLASS_NAME
SEOCLSNAME
USE_DATEFORMAT
USE_EXTPROCESS
USE_DETCALENDAR
USE_DETDURATION
USE_DETTZONE
USE_DETEXTPROC
USE_DETSUBNET
USE_DETLOCATION
USE_MAPPROPERTY
USE_DETPROPERTY
/SCMB/TSCIMAP
PK,FK1 CLIENT
PK SCHEME
PK DATA_NAME
PK PRIO
PROP_NAME
FK1 CLASS_NAME
/SCMB/TSCMAST
PK,FK1 CLIENT
PK SCHEME
PK DATA_NAME
PK PRIO
FK1 CLASS_NAME
/SCMB/TSCLASST
PK,FK1 CLIENT
PK LANGUAGE
PK,FK1 CLASS_NAME
TEXT_CLASS

Fig. 6B

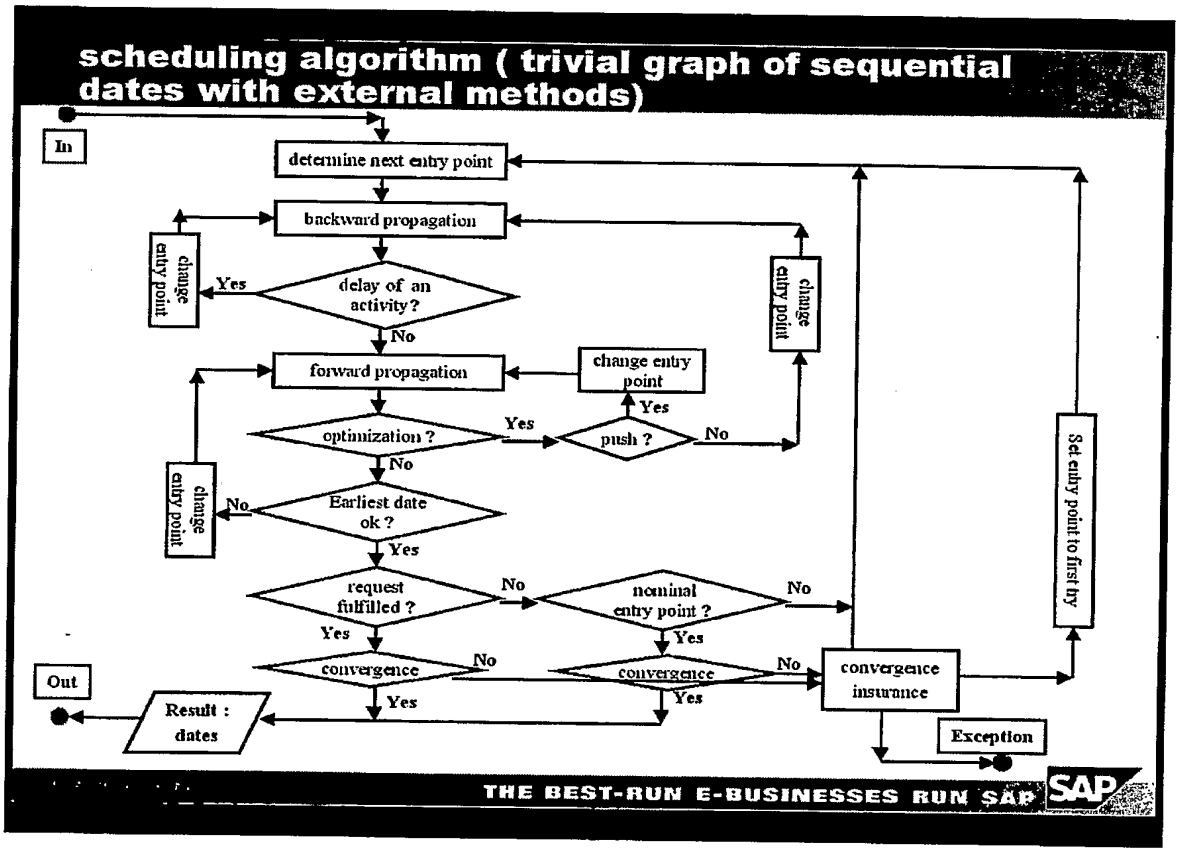

scheduling algorithm ( trivial graph of sequential dates with external methods)
In
determine next entry point
backward propagation
change entry point
Yes
delay of an activity ?
No
change entry point
forward propagation
change entry point
Yes
optimization ?
Yes
push ?
No
No
change entry point
No
Earliest date ok ?
Yes
request fulfilled ?
No
nominal entry point ?
No
Yes
Yes
convergence
No
convergence
No
convergence insurance
Set entry point to first try
Out
Result : dates
Yes
Yes
Exception
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 7A

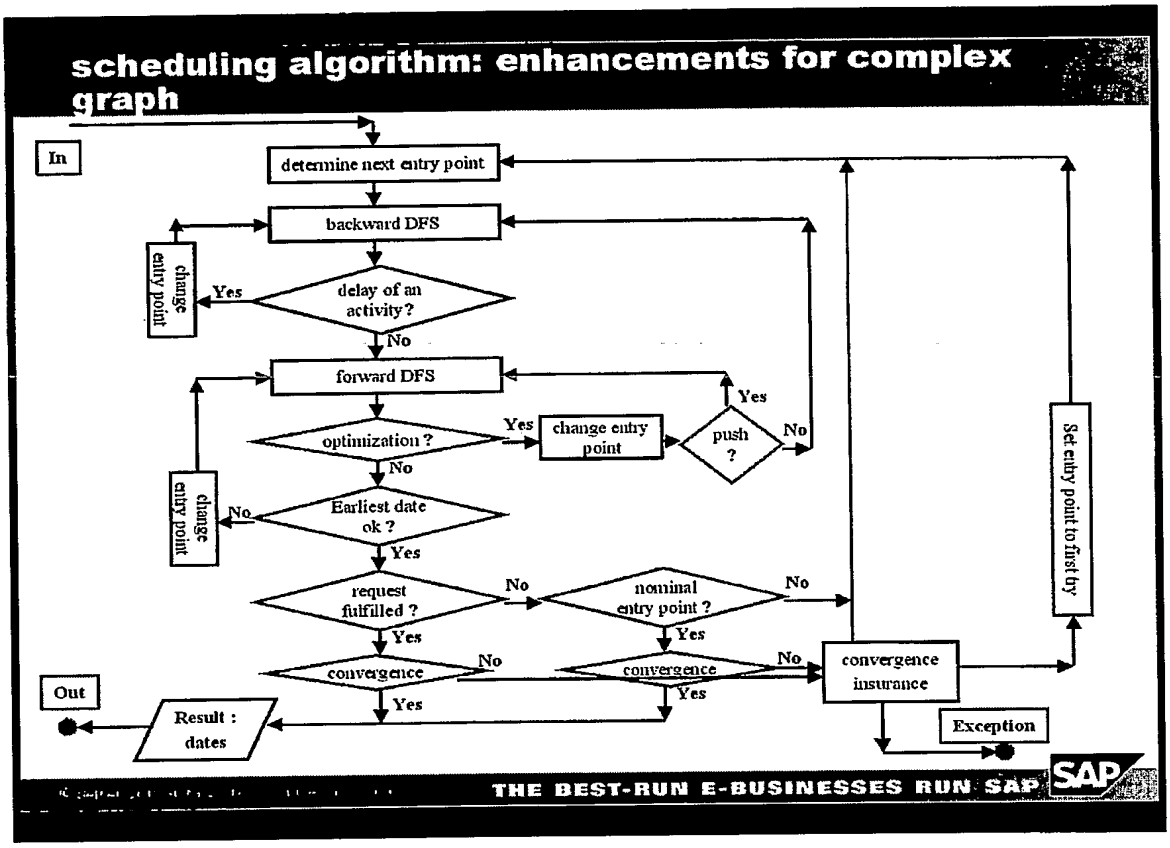

scheduling algorithm: enhancements for complex graph
In
determine next entry point
backward DFS
change entry point
Yes
delay of an activity?
No
forward DFS
Yes
optimization ?
Yes
change entry point
push ?
No
No
change entry point
No
Earliest date ok ?
Yes
request fulfilled ?
No
nominal entry point ?
No
Yes
Yes
convergence
No
convergence
No
convergence insurance
Set entry point to first try
Yes
Yes
Out
Result : dates
Exception
THE BEST-RUN E-BUSINESSES RUN SAP
SAP

Fig. 7B

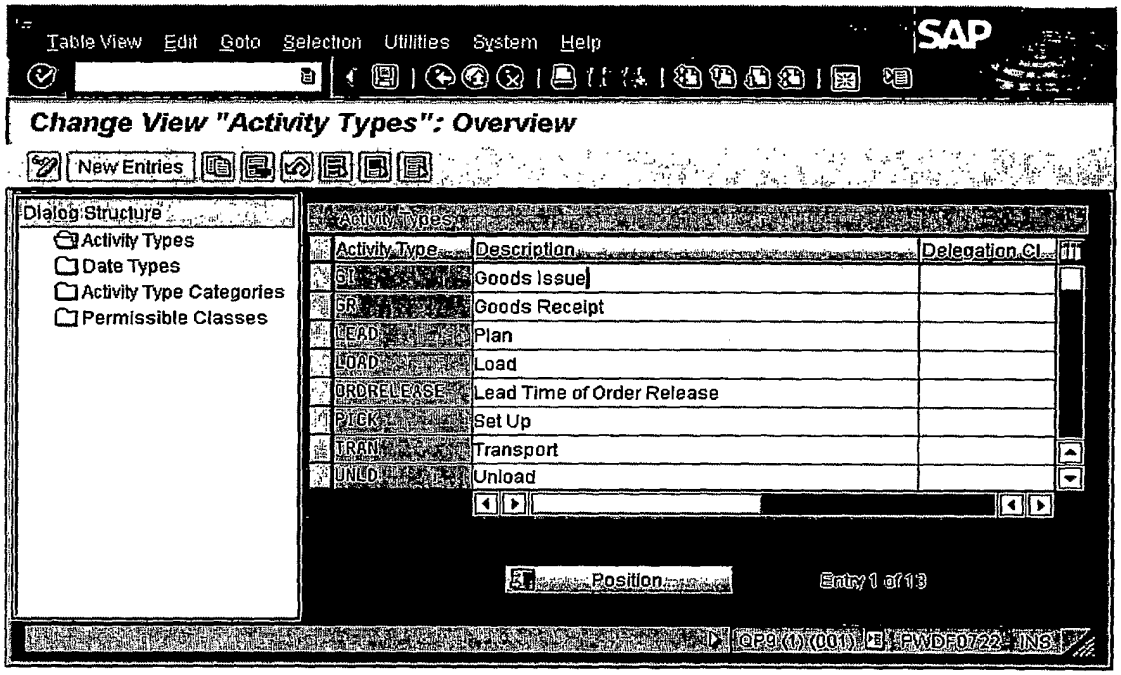


Table View Edit Goto Selection Utilities System Help
SAP
Change View "Activity Types": Overview
New Entries
Dialog Structure
Activity Types
Date Types
Activity Type Categories
Permissible Classes
Activity Types
Activity Type
Description
Delegation Cl...
Goods Issue
Goods Receipt
LEAD
Plan
LOAD
Load
ORDRELEASE
Lead Time of Order Release
PICK
Set Up
TRAN
Transport
UNLO
Unload
Position...
Entry 1 of 13
QP9 (1) (001)
FWDF0722
INS

Fig. 8A

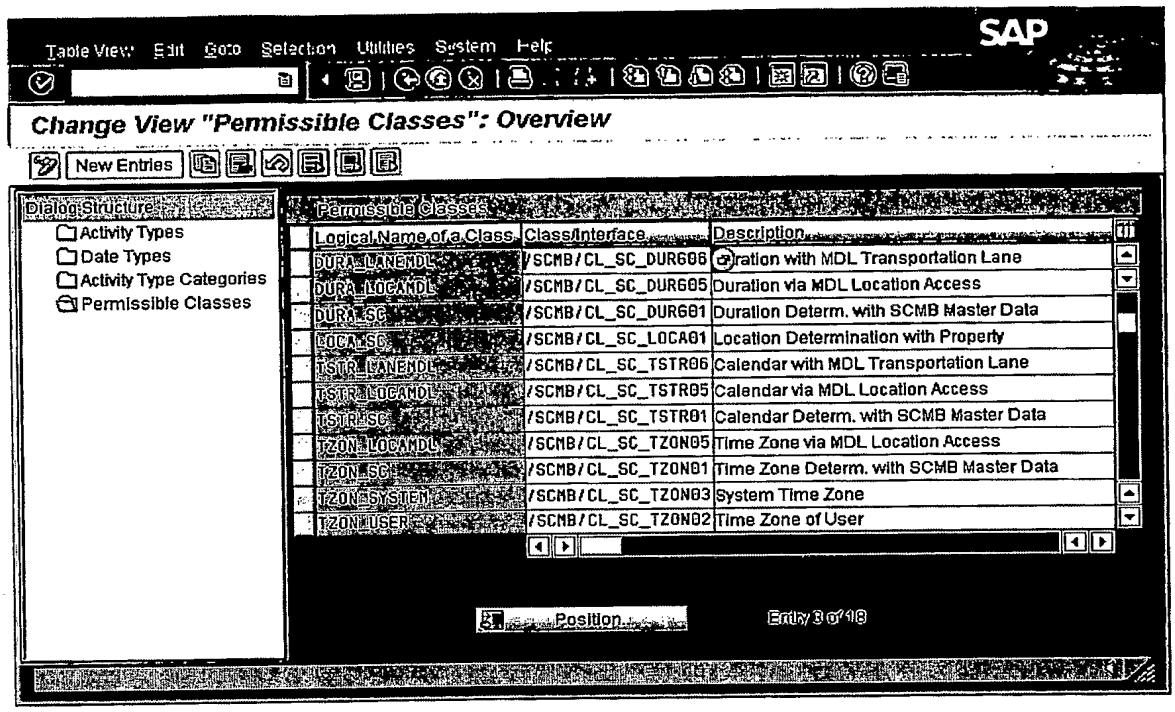


Table View Edit Goto Selection Utilities System Help
SAP
Change View "Permissible Classes": Overview
New Entries
Dialog Structure
Activity Types
Date Types
Activity Type Categories
Permissible Classes
Permissible Classes
Logical Name of a Class
Class/Interface
Description
DURA_LANEMDL /SCMB/CL_SC_DURG06 Duration with MDL Transportation Lane
DURA_LOCANDL /SCMB/CL_SC_DURG05 Duration via MDL Location Access
DURA_SC /SCMB/CL_SC_DURG01 Duration Determ. with SCMB Master Data
LOCA_SC /SCMB/CL_SC_LOCA01 Location Determination with Property
TSTR_LANEMDL /SCMB/CL_SC_TSTR06 Calendar with MDL Transportation Lane
TSTR_LOCANDL /SCMB/CL_SC_TSTR05 Calendar via MDL Location Access
TSTR_SC /SCMB/CL_SC_TSTR01 Calendar Determ. with SCMB Master Data
TZON_LOCANDL /SCMB/CL_SC_TZON05 Time Zone via MDL Location Access
TZON_SC /SCMB/CL_SC_TZON01 Time Zone Determ. with SCMB Master Data
TZON_SYSTEM /SCMB/CL_SC_TZON03 System Time Zone
TZON_USER /SCMB/CL_SC_TZON02 Time Zone of User
Position...
Entry 9 of 18

Fig. 8B

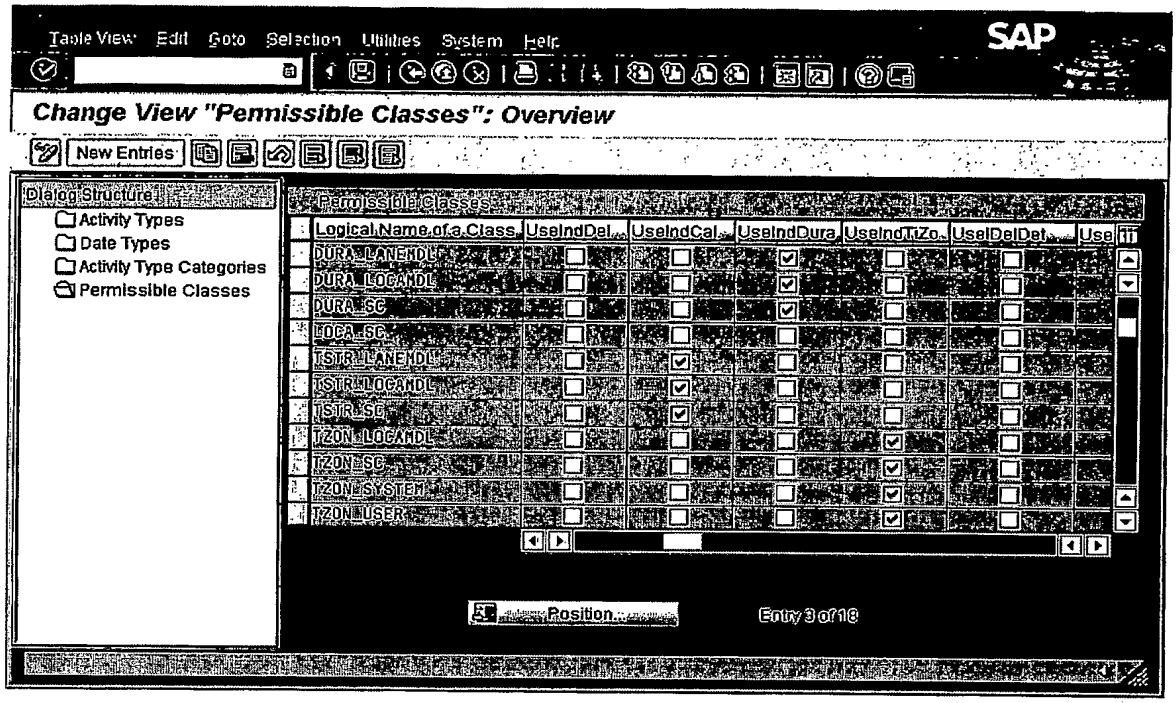


Table View Edit Goto Selection Utilities System Help
SAP
Change View "Permissible Classes": Overview
New Entries
Dialog Structure
Activity Types
Date Types
Activity Type Categories
Permissible Classes
Permissible Classes
Logical Name of a Class
UseIndDel
UseIndCal
UseIndDura
UseIndTiZo
UseIndDelDet
Use
DURA_LANEMDL
DURA_LOCAMDL
DURA_SC
LOCA_SC
TSTR_LANEMDL
TSTR_LOCAMDL
TSTR_SC
TZON_LOCAMDL
TZON_SC
TZON_SYSTEM
TZON_USER
Position...
Entry 9 of 18

Fig. 8C

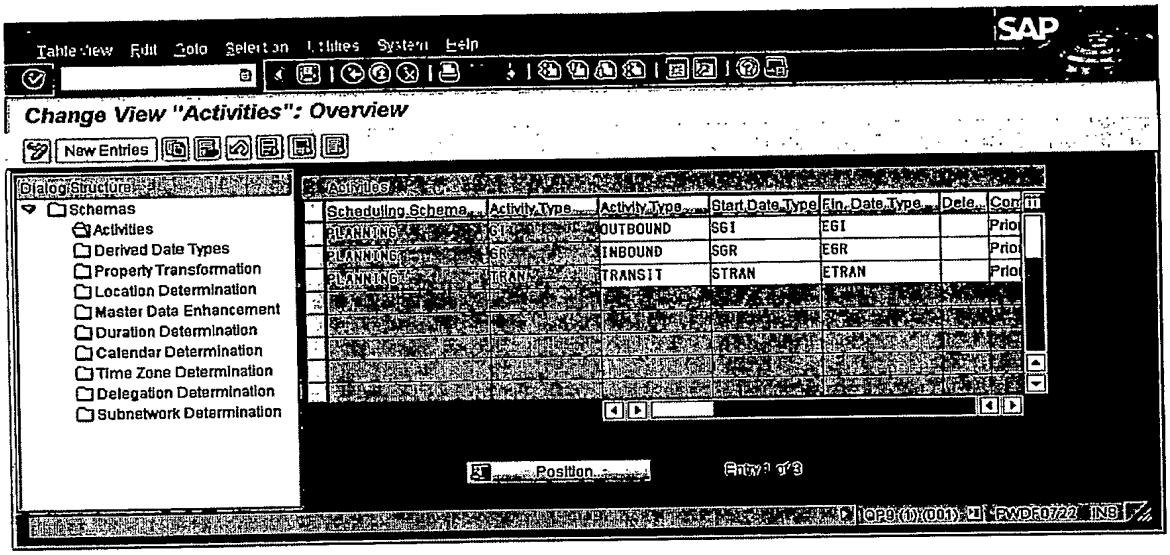


Table View Edit Goto Selection Utilities System Help
SAP
Change View "Activities": Overview
New Entries
Dialog Structure
Schemas
Activities
Derived Date Types
Property Transformation
Location Determination
Master Data Enhancement
Duration Determination
Calendar Determination
Time Zone Determination
Delegation Determination
Subnetwork Determination
Activities
Scheduling Schema
Activity Type
Activity Type
Start Date Type
Fin. Date Type
Dele...
Con
PLANNING
OUTBOUND
SGI
EGI
Prio
PLANNING
INBOUND
SGR
EGR
Prio
PLANNING
TRANSIT
STRAN
ETRAN
Prio
Position...
Entry 1 of 3
FWDR0722
INS

Fig. 8D

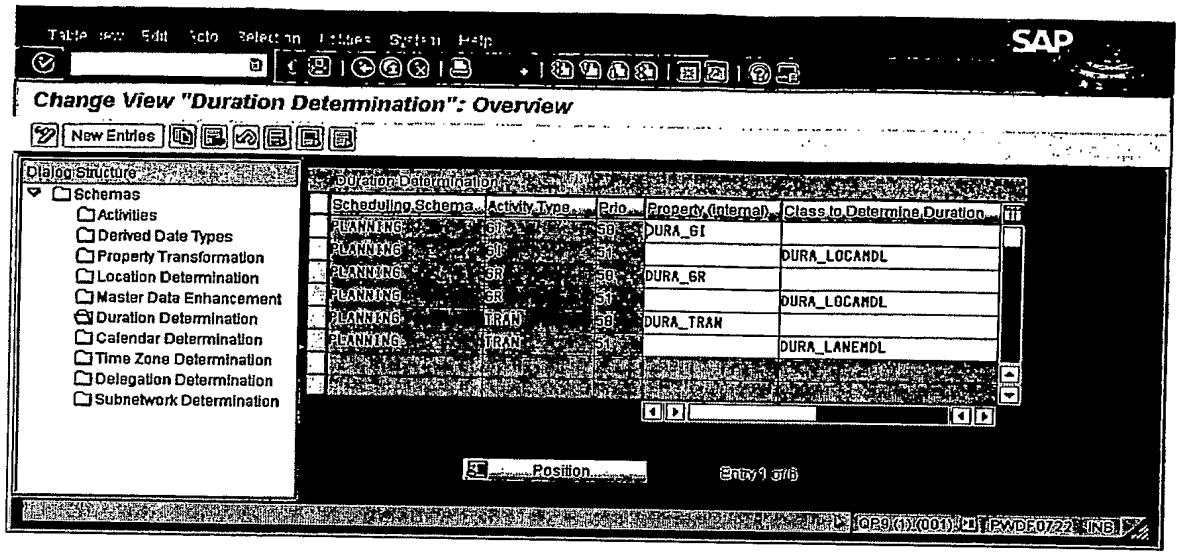

Change View "Duration Determination": Overview
New Entries
Dialog Structure
Schemas
Activities
Derived Date Types
Property Transformation
Location Determination
Master Data Enhancement
Duration Determination
Calendar Determination
Time Zone Determination
Delegation Determination
Subnetwork Determination
Duration Determination
Scheduling Schema
Activity Type
Prio
Property (internal)
Class to Determine Duration
PLANNING
6I
50
DURA_6I
51
DURA_LOCAMDL
6R
50
DURA_6R
51
DURA_LOCAMDL
TRAN
50
DURA_TRAN
51
DURA_LANEMDL
Position...
Entry 1 of 6

Fig. 8E

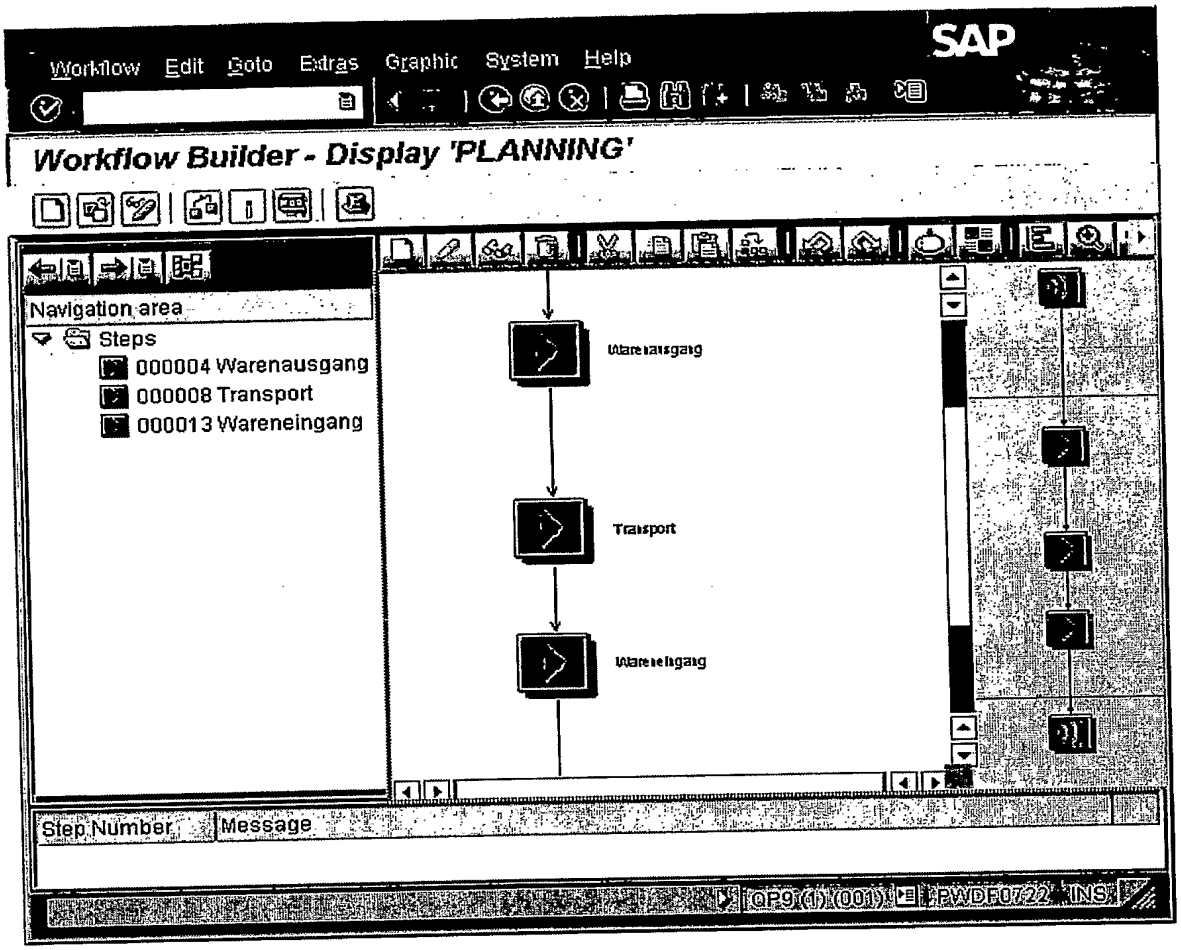

Workflow Edit Goto Extras Graphic System Help
SAP
Workflow Builder - Display 'PLANNING'
Navigation area
Steps
000004 Warenausgang
000008 Transport
000013 Wareneingang
Warenausgang
Transport
Wareneingang
Step Number
Message

Fig. 9

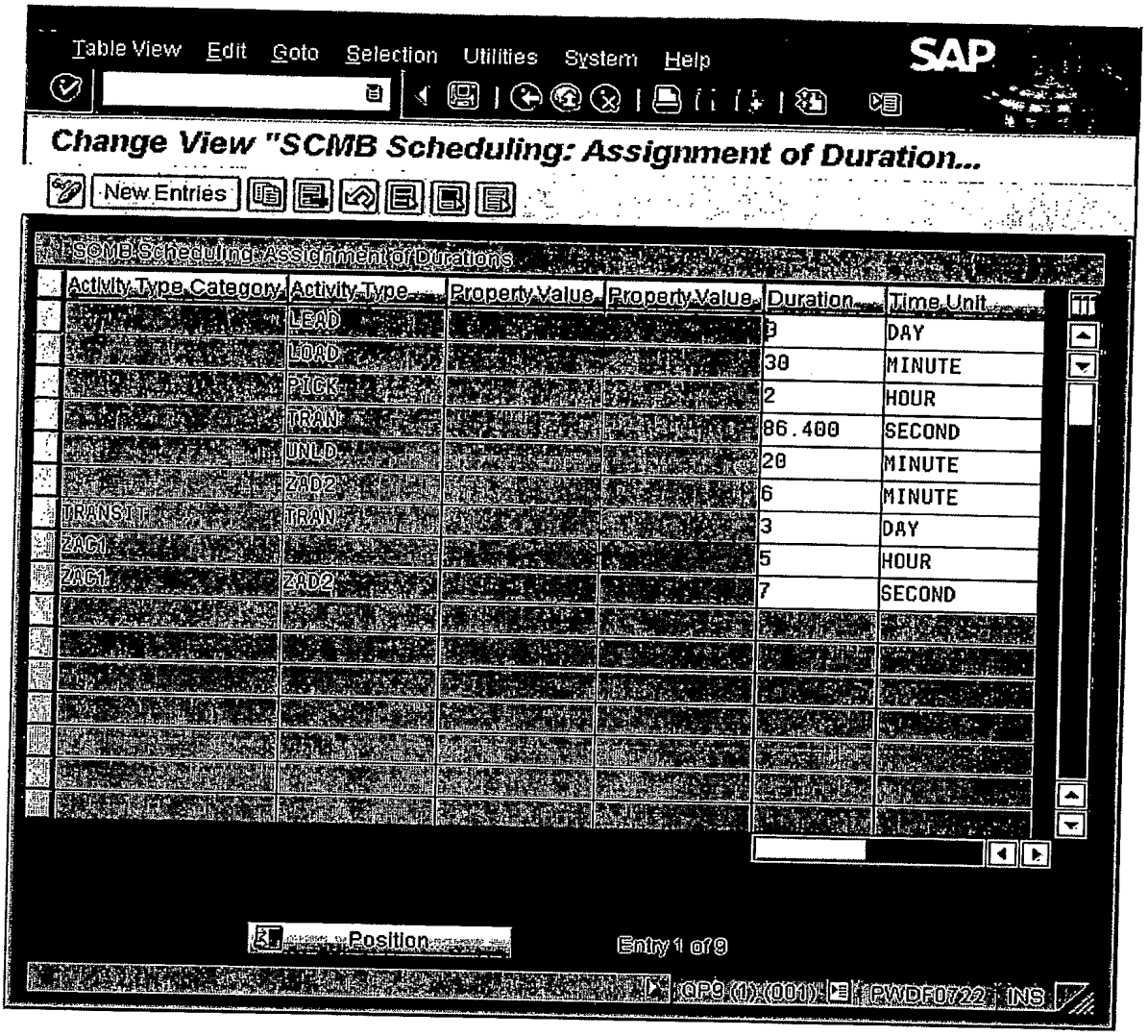


Table View Edit Goto Selection Utilities System Help
SAP
Change View "SCMB Scheduling: Assignment of Duration...
New Entries
SCMB Scheduling: Assignment of Durations
Activity Type Category
Activity Type
Property Value
Property Value
Duration
Time Unit
LEAD 3 DAY
LOAD 30 MINUTE
PICK 2 HOUR
TRAN 86.400 SECOND
UNLD 20 MINUTE
ZAD2 6 MINUTE
TRANSIT TRAN 3 DAY
ZAC1 5 HOUR
ZAC1 ZAD2 7 SECOND
Position
Entry 1 of 9
QP9 (1) (001)
PWDF0722
INS

Fig. 10

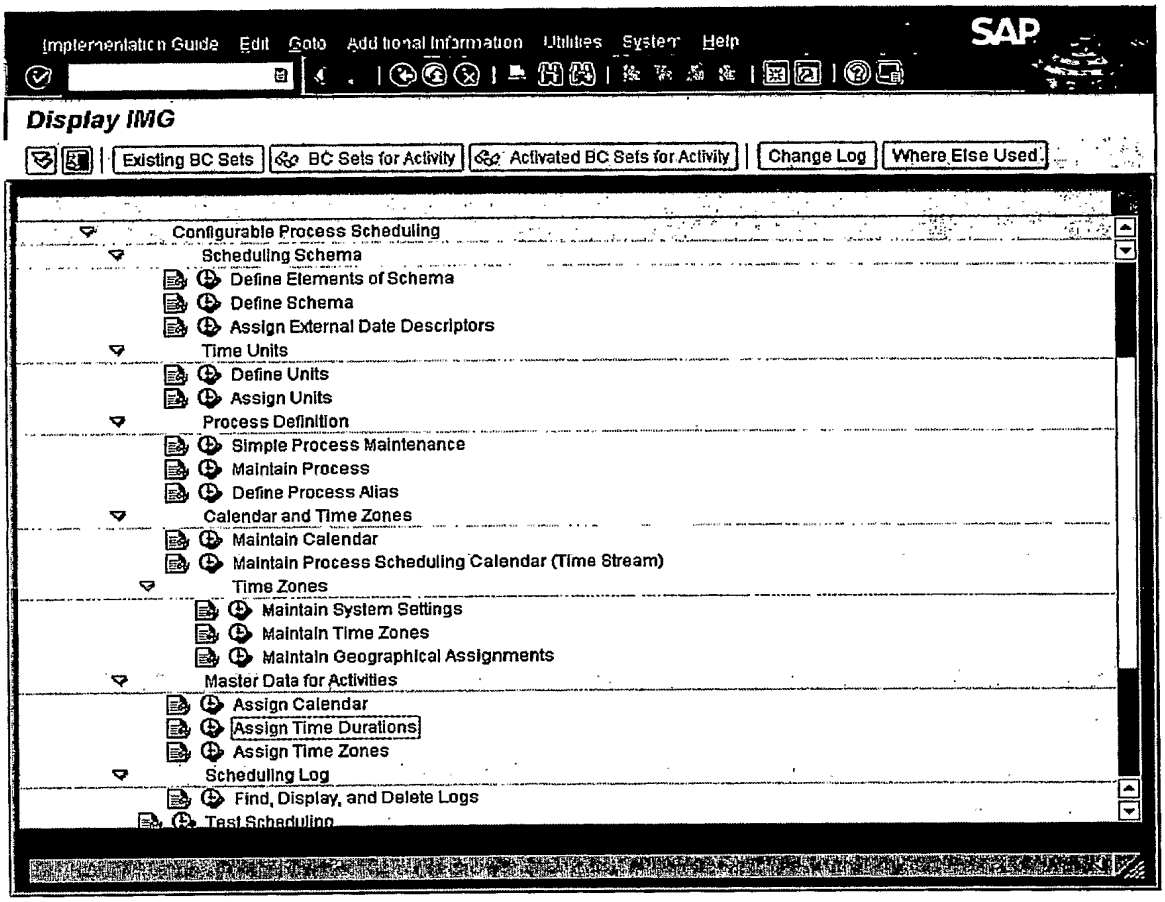

Implementation Guide Edit Goto Additional Information Utilities System Help
SAP
Display IMG
Existing BC Sets
BC Sets for Activity
Activated BC Sets for Activity
Change Log
Where Else Used
Configurable Process Scheduling
Scheduling Schema
Define Elements of Schema
Define Schema
Assign External Date Descriptors
Time Units
Define Units
Assign Units
Process Definition
Simple Process Maintenance
Maintain Process
Define Process Alias
Calendar and Time Zones
Maintain Calendar
Maintain Process Scheduling Calendar (Time Stream)
Time Zones
Maintain System Settings
Maintain Time Zones
Maintain Geographical Assignments
Master Data for Activities
Assign Calendar
Assign Time Durations
Assign Time Zones
Scheduling Log
Find, Display, and Delete Logs
Test Scheduling

Fig. 11

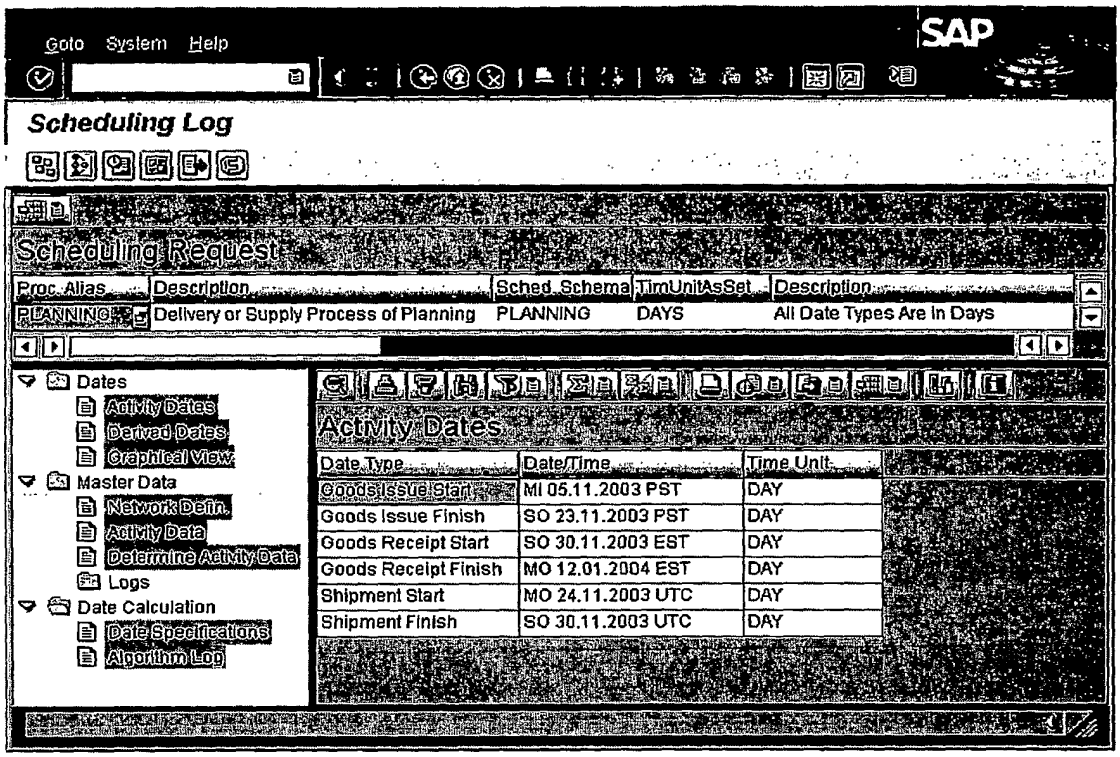

Goto System Help
SAP
Scheduling Log
Scheduling Request
Proc. Alias
Description
Sched. Schema
TimUnitAsSet
Description
PLANNING
Delivery or Supply Process of Planning
PLANNING
DAYS
All Date Types Are In Days
Dates
Activity Dates
Derived Dates
Graphical View
Master Data
Network Defin.
Activity Data
Determine Activity Data
Logs
Date Calculation
Date Specifications
Algorithm Log
Activity Dates
Date Type
Date/Time
Time Unit
Goods Issue Start
MI 05.11.2003 PST
DAY
Goods Issue Finish
SO 23.11.2003 PST
DAY
Goods Receipt Start
SO 30.11.2003 EST
DAY
Goods Receipt Finish
MO 12.01.2004 EST
DAY
Shipment Start
MO 24.11.2003 UTC
DAY
Shipment Finish
SO 30.11.2003 UTC
DAY

Fig. 12A

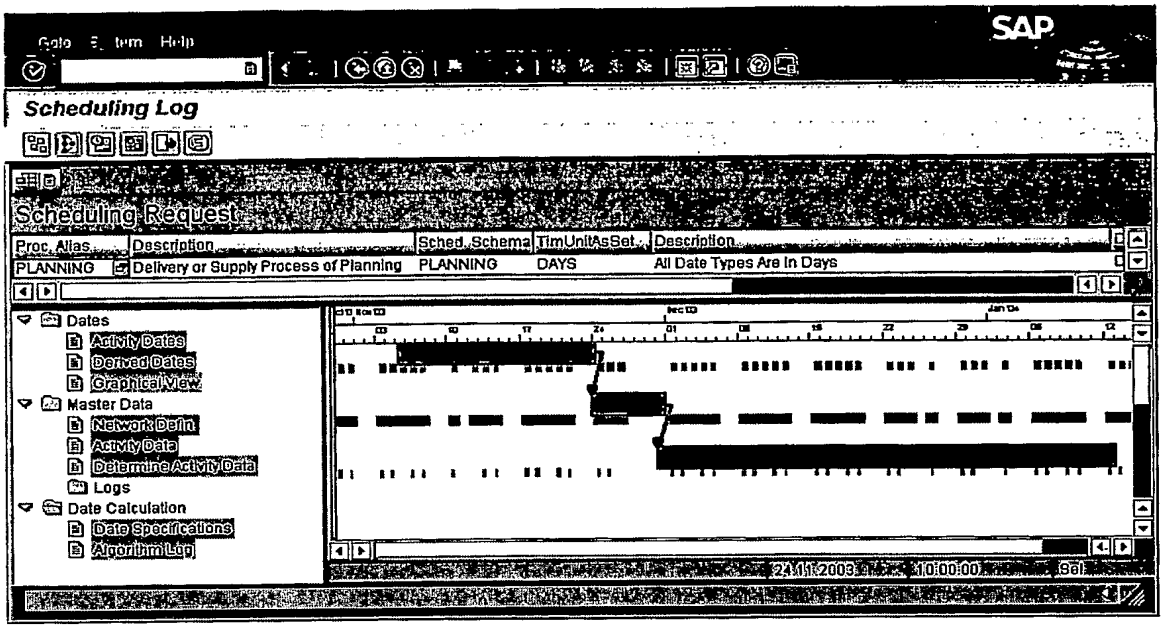

Goto Extras Help
SAP
Scheduling Log
Scheduling Request
Proc. Alias
Description
Sched. Schema
TimUnitAsSet
Description
PLANNING
Delivery or Supply Process of Planning
PLANNING
DAYS
All Date Types Are In Days
Dates
Activity Dates
Derived Dates
Graphical View
Master Data
Network Defin.
Activity Data
Determine Activity Data
Logs
Date Calculation
Date Specifications
Algorithm Log
24.11.2003
10:00:00

Fig. 12B

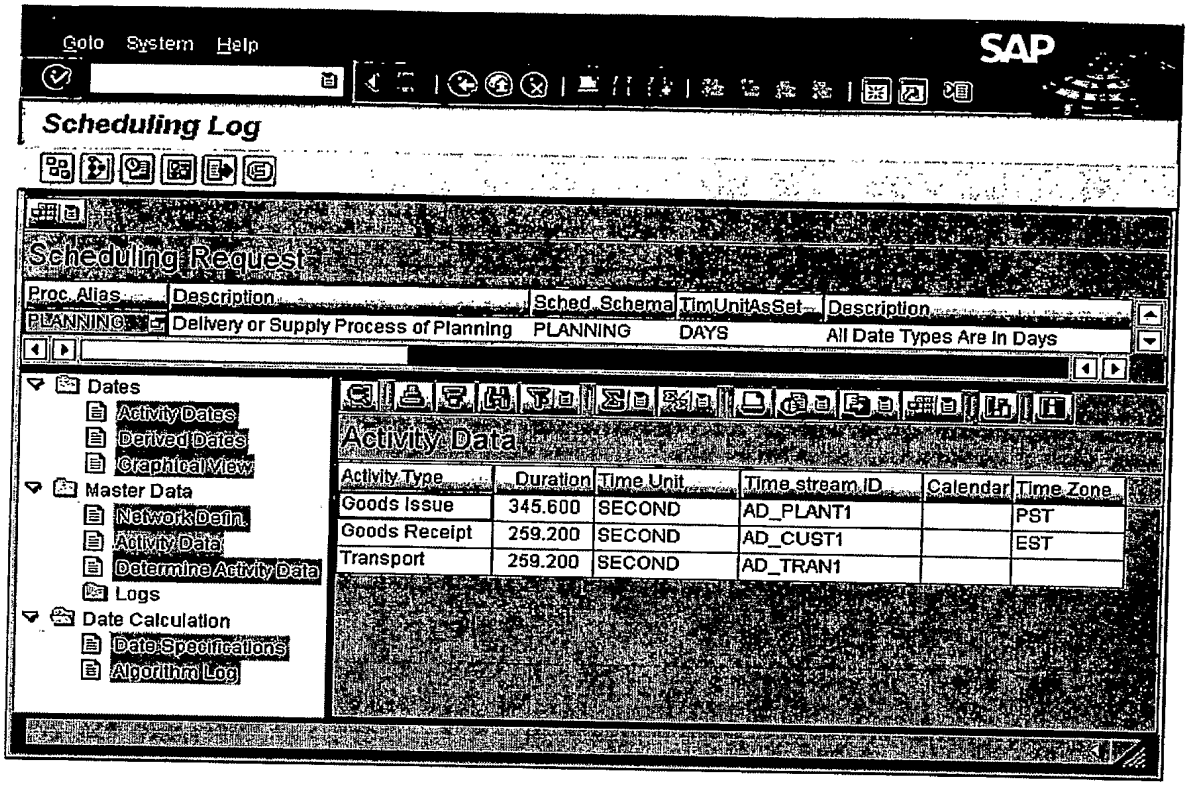

Goto System Help
SAP
Scheduling Log
Scheduling Request
Proc. Alias
Description
Sched. Schema
TimUnitAsSet
Description
PLANNING
Delivery or Supply Process of Planning
PLANNING
DAYS
All Date Types Are In Days
Dates
Activity Dates
Derived Dates
Graphical View
Master Data
Network Defin.
Activity Data
Determine Activity Data
Logs
Date Calculation
Date Specifications
Algorithm Log
Activity Data
Activity Type
Duration
Time Unit
Time stream ID
Calendar
Time Zone
Goods Issue
345.600
SECOND
AD_PLANT1
PST
Goods Receipt
259.200
SECOND
AD_CUST1
EST
Transport
259.200
SECOND
AD_TRAN1

Fig. 12C